NAVAL POSTGRADUATE SCHOOL
Monterey, California

DISSERTATION

SYSTEM INTERDICTION AND DEFENSE

by

Eitan Israeli

March 1999

Dissertation Advisor: R. Kevin Wood

Approved for public release; distribution is unlimited.

19990419 070

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 1999	3. REPORT TYPE AND DATES COVERED Doctoral Dissertation

4. TITLE AND SUBTITLE **SYSTEM INTERDICTION AND DEFENSE**	5. FUNDING NUMBERS
6. AUTHOR(S) Israeli, Eitan	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

We study the problem of interdicting components of an adversary's system, e.g., a war-time economy, a transportation network, etc. Basic techniques are developed and illustrated with a simple network interdiction problem, "maximizing the shortest path" (MXSP). In MXSP, an interdictor wishes to employ limited interdiction resources as effectively as possible to slow an adversary in moving between two network nodes. "Interdiction" destroys a network arc entirely or increases its effective length through an attack. This bi-level, max-min problem is formulated as a mixed-integer program (MIP), but unique decomposition algorithms are developed to solve the problem more efficiently than standard branch and bound. One algorithm is essentially Benders decomposition with special integrality cuts for the master problem. A second algorithm uses a new set-covering master problem, and a third is a hybrid of the first two. We extend our techniques (*i*) to solve general system-interdiction problems, some of which cannot be formulated as MIPs, (*ii*) to solve system-defense problems where critical system components must be identified and hardened against interdiction, and (*iii*) to solve interdiction problems with uncertain interdiction success. We report computational experience for MXSP, a shortest-path network-defense problem and MXSP with uncertain interdiction success.

14. SUBJECT TERMS Network, Interdiction, Defense, Benders Decomposition, Stochastic Programming, Set Covering	15. NUMBER OF PAGES 144
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFI- CATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

i

Approved for public release; distribution is unlimited

SYSTEM INTERDICTION AND DEFENSE

Eitan Israeli
Major, Israeli Air-Force
B.Sc., Hebrew University of Jerusalem, 1985
M.Sc., Tel-Aviv University, 1990

Submitted in partial fulfillment of the
requirements for the degree of

## DOCTOR OF PHILOSOPHY IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
March 1999
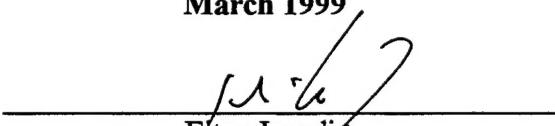
Author: _____
Eitan Israeli

Approved by: _____
R. Kevin Wood
Professor of Operations Research
Dissertation Supervisor

_____   _____
Gerald G. Brown   Alan R. Washburn
Professor of Operations Research   Professor of Operations Research
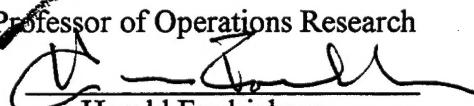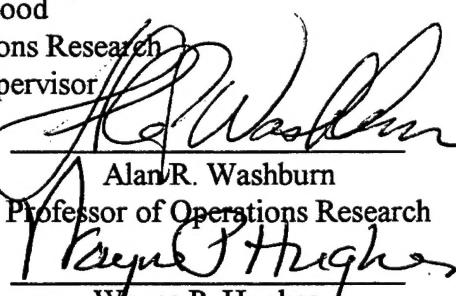
_____   _____
Harold Fredricksen   Wayne P. Hughes
Professor of Mathematics   Adjunct Professor of Operations Research
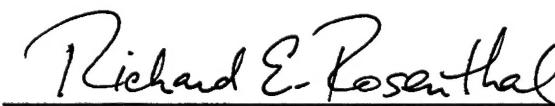
Approved by: _____
Richard E. Rosenthal, Chair,
Department of Operations Research
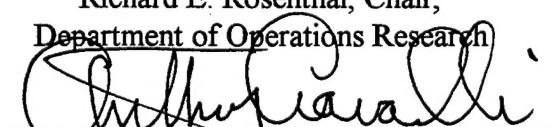
Approved by: _____
Anthony Ciavarelli,
Acting Associate Provost for Instruction

iii

ABSTRACT

We study the problem of interdicting components of an adversary's system, e.g., a war-time economy, a transportation network, etc. Basic techniques are developed and illustrated with a simple network interdiction problem, "maximizing the shortest path" (MXSP). In MXSP, an interdictor wishes to employ limited interdiction resources as effectively as possible to slow an adversary in moving between two network nodes. "Interdiction" destroys a network arc entirely or increases its effective length through an attack. This bi-level, max-min problem is formulated as a mixed-integer program (MIP), but unique decomposition algorithms are developed to solve the problem more efficiently than standard branch and bound. One algorithm is essentially Benders decomposition with special integrality cuts for the master problem. A second algorithm uses a new set-covering master problem, and a third is a hybrid of the first two. We extend our techniques (*i*) to solve general system-interdiction problems, some of which cannot be formulated as MIPs, (*ii*) to solve system-defense problems where critical system components must be identified and hardened against interdiction, and (*iii*) to solve interdiction problems with uncertain interdiction success. We report computational experience for MXSP, a shortest-path network-defense problem and MXSP with uncertain interdiction success.

TABLE OF CONTENTS

ACKNOWLEDGEMENT

I. INTRODUCTION

This doctoral dissertation investigates what we call "The System Interdiction Problem," (SI), and "The System Defense Problem," (SD). In SI, an "adversary" tries to maximize the utility of his system (modeled as general linear or integer programs), and an "interdictor," with limited assets, tries to minimize that maximum by limiting the adversary's feasible actions, or by increasing the cost associated with his activities. As in a zero-sum Static Stackelberg Game (see Simaan and Cruz, 1973, for the definition of Stackelberg strategy), we assume that the interdictor (*leader*) first chooses his actions and only after that the adversary (*follower*) decides how to operate the system, as best possible, given the effects of the interdiction. SD extends that methodology to plan effective "hardening" (defense) of a system to minimize the effects of subsequent interdiction. In SD, the players change sides: The leader is the system user who first chooses his defense actions and only then the interdictor, now the follower, chooses his interdiction plan.

Throughout this work, we have a special interest in the problem of interdicting (or defending) a road or other transportation network in order to maximize the post-interdiction shortest-path length between two specified nodes. In this problem, a "network user" wishes to traverse a path of minimum length (or minimum time, minimum cost, etc.) between two specified nodes, s and t, in a directed network. But, by first attacking the network using limited resources, an interdictor can destroy certain arcs, or increase the effective length of certain arcs, and thereby increase the minimum length in the a priori network.

(a)   *Maximizing the Shortest Path* (MXSP) is the interdictor's max-min problem: Subject to a limited interdiction budget, interdict arcs in a network so as to maximize the shortest-path length between nodes s and t.

(b)   *Defending the Shortest Path* (DSP) is the network user's min-max-min problem: Subject to a limited defense budget, harden arcs against interdiction so as to minimize the post-interdiction shortest path, given that the interdictor will optimize his interdiction plan with knowledge of which arcs are hardened. Hardened arcs are assumed invulnerable.

In the rest of this chapter, we formulate SI and SD as mathematical programming problems, and motivate some new approaches for solving those problems.

A.   SYSTEM INTERDICTION

The system interdiction problem is a generalization of network interdiction problems, which have received considerable attention over the years. First were the military applications, like interdiction of ground-forces transportation (e.g., Ghare, Montgomery and Turner 1971, McMasters and Mustin 1970, Golden 1978, Fulkerson and Harding 1977), and lately drug interdiction efforts have triggered more research (e.g., Wood 1993, Washburn and Wood 1994). Today, military and civilian systems are becoming even more complicated and interdependent, so interest in interdiction of "general systems" arises, too (Chern and Lin 1995).

The system interdiction problem is a model for the following scenario: Two opposing forces, a leader (interdictor) and a follower (adversary), are involved in a

warlike conflict. We assume that the follower operates a "system," with its optimal operation represented adequately by the solution to a linear program. (Later we extend our results to interdiction of systems represented by more general optimization problems.) Thus, the follower's problem, with no interdiction, is simply

$\max\{\mathbf{c}^T\mathbf{y} \,|\, A\mathbf{y} \le \mathbf{b}, 0 \le \mathbf{y} \le \mathbf{u}\}$ where $\mathbf{c}, \mathbf{y}, \mathbf{u} \in \Re^n$, $\mathbf{b} \in \Re^m, A \in \Re^{m \times n}$ and $\mathbf{c}^T$ is the

transpose of the column vector $\mathbf{c}$ (we assume that the follower's problem is feasible). On the other hand, the interdictor tries to minimize the follower's objective value by preventing the use of some of the follower's possible activities, indexed by j. Let the feasible set for the leader be $X = \{\mathbf{x} \in \{0,1\}^n \,|\, R\mathbf{x} \le \mathbf{r}\}$. $x_j = 1$ means that activity j is interdicted and $R\mathbf{x} \le \mathbf{r}$ represents the restrictions on interdiction resources (we assume that X is not empty).

The system interdiction problem finds the optimal interdiction strategy $\mathbf{x}^*$ for the leader. The optimal solution for the follower given $\mathbf{x}^*$, denoted by $\mathbf{y}^*$, is not particularly important. Two special instances of SI are:

(a) Interdiction of a max-flow network system (e.g., Ghare, Montgomery and Turner 1971, Wood 1993) is a situation where the follower maximizes flow through a capacitated network, while the leader, with limited interdiction resources, can break some of the network's arcs (a broken arc has no capacity), and

(b) The *k-most-vital-arcs problem* (Corely and Shaw 1982, Malik *et al.* 1989) is a special case of MXSP where the interdictor seeks to destroy exactly k arcs to interdict the network most effectively.

Let $U = \text{diag}(\mathbf{u})$. Then, *The Linear System-Interdiction Problem* (LSIP) is defined to be the following leader's problem:

[LSIP] $\displaystyle \min_{\mathbf{x} \in X} \; \max_{\mathbf{y} \in Y(\mathbf{x})} \; \mathbf{c}^T \mathbf{y}$

where $X \quad = \left\{ \mathbf{x} \in \{0,1\}^n \;\middle|\; R\mathbf{x} \le \mathbf{r} \right\}$, and

$Y(\mathbf{x}) \; = \left\{ \mathbf{y} \;\middle|\; A\mathbf{y} \le \mathbf{b}, \; \mathbf{0} \le \mathbf{y} \le U(\mathbf{1} - \mathbf{x}) \right\}$

The follower uses activity j at level y_j. By interdicting activity j, the leader changes the upper bound on y_j from u_j to 0, i.e., forces the follower to accept $y_j = 0$ (we assume that $Y(\mathbf{x})$ is not empty for all feasible $\mathbf{x} \in X$).

Remark: In **Appendix B** we show that [LSIP] is equivalent to a more general system interdiction problem where $Y(\mathbf{x}) = \left\{ \mathbf{y} \middle| A\mathbf{y} + B\mathbf{x} \le \mathbf{b} \right\}$ and where the leader can change the cost of the follower's activities as well. This general case allows any single interdiction by the leader to affect one or more of the follower's possible activities and available resources. For instance, in MXSP, one interdiction attempt might increase the length of several arcs and/or delete one or more nodes from the network.

Formulation [LSIP] is a structured case of the *Bi-Level Mixed Integer Programming Problem* (BLMIP). (See Ben-Ayed 1993 for an introduction to BLMIPs.) However, the general BLMIP does not assume a max-min conflict as in LSIP, and the objective functions of the leader and the follower may have much in common in the BLMIP, rather than being in direct opposition. For instance, the leader and follower may represent two levels of decision makers in the same company, and therefore they have similar, although not identical, goals.

In **Appendix A** existing algorithms for BLMIPs are explored, and it is shown that none of those algorithms is appropriate for large instances of LSIP. Some of the algorithms (Bard and Moore 1992, Wen and Yang 1990) use a *positive approach*, which means that they work better when there is strong correlation between the leader's and follower's objective functions. Such algorithms are likely to be inefficient when applied to max-min problems such as LSIP. Two other BLMIP algorithms (Moore and Bard 1990, Vicente *et al.* 1996) may not be directly positive, but they rely on solving bi-level linear programs (BLLPs) which are nominally solved using a positive approach. In fact, only three exact algorithms for BLLPs have been tested on relatively large problems (Bard and Moore 1990, Hansen *et al.* 1992, Judice and Faustino 1992), and all of these algorithms use a positive approach. A few exact algorithms (Vaish and Shetty 1977, Anandalingam and Apprey 1991, Onal, 1993) use what appears to be a *non-positive approach*, but none of these algorithms have been tested on large problems. Finally, we note that no extant algorithm in the bi-level arena, exact or heuristic, is designed to take advantage of the special max-min structure of LSIP, or the special shortest-path structure of MXSP.

B.   REFORMULATION

Formulation [LSIP] is difficult to solve, as we shall see, and a reformulation is needed. In order to see the difficulties with [LSIP], notice first that being able to solve [LSIP] when x is continuous would be useful, because:

(a)   We would be able to use a branch-and-bound procedure to solve [LSIP].

(b)   The relaxed problem has a solution at an extreme point of the convex hull of X, $C(X)$ (Bazaraa, et. al., 1993). Hence, if we relax the binary constraints $\mathbf{x} \in \{0,1\}^n$ so that $\mathbf{0} \leq \mathbf{x} \leq \mathbf{1}$, and all the extreme points of $C(X)$ happen to be in X, then we can relax the binary constraints and still be guaranteed an optimal (binary) solution. For instance, this would be the case when $X = \{\ \mathbf{x} \in \{0,1\}^n,\ \mathbf{1x} \leq r_0\}$).

However, solving the relaxed problem isn't easy in general. Let $f_1(\mathbf{x}) \equiv \max\limits_{\mathbf{y} \in Y(\mathbf{x})} \mathbf{c}^T \mathbf{y}$ so that formulation [LSIP] is equivalent to $\min\limits_{\mathbf{x} \in X} f_1(\mathbf{x})$. When we relax the binary constraints in X, $f_1(\mathbf{x})$ is a concave function, because the choice of $\mathbf{x}$ changes only the right-hand side of the follower's LP. Hence, we may have local optima. In fact, the problem [LSIP] is extremely difficult to solve and even the relaxed problem is strongly NP-hard (Hansen, Jaumard and Savard 1992).

Example 1.1

To illustrate the "bad behavior" of the relaxation of [LSIP], let the follower's problem be represented by:

[EX1] $\max\limits_{\mathbf{y}}$ $y_1 + y_2$

          s.t. $y_1 \qquad\qquad \leq 5$

                    $y_2 \quad \leq 5$

                    $y_1 - y_2 \quad \leq 4$

                    $-y_1 + y_2 \quad \leq 3$

                    $y_1 \geq 0,\ y_2 \geq 0,$

The system's value (without any interdiction) is 10. Assume that the leader can interdict

either y_1 or y_2; of course, he wishes to minimize the maximum value of $y_1 + y_2$ after interdiction. If the leader interdicts y_1, the follower solves

[EX1a]
$$\max_{y} \quad y_1 + y_2$$

$$
\begin{aligned}
\text{s.t.} \quad y_1 & \leq 0 \\
y_2 & \leq 5 \\
y_1 - y_2 & \leq 4 \\
-y_1 + y_2 & \leq 3 \\
y_1 \geq 0, \ y_2 & \geq 0,
\end{aligned}
$$

and hence obtains a value of 3. In the same way, if the leader interdicts y_2, the value of the system is 4. In this simple example, the optimal solution for the leader is to interdict y_1. Now, let's see what happens when we relax the binary constraint on the interdiction variables. Let y_1 be interdicted by $1 - p$ and y_2 be interdicted by p, $0 \leq p \leq 1$. The parametric problem the leader solves is:

[EX1b]
$$\min_{0 \leq p \leq 1} f_1(p)$$

where $f_1(p) = \max_{y} \ y_1 + y_2$

$$
\begin{aligned}
\text{s.t.} \quad y_1 & \leq 5p \\
y_2 & \leq 5(1-p) \\
y_1 - y_2 & \leq 4 \\
-y_1 + y_2 & \leq 3 \\
y_1 \geq 0, \ y_2 & \geq 0,
\end{aligned}
$$

and $f_1(p)$ has a global minimum at $p = 0$ and a local minimum at $p = 1$. (See **Figure 1.1.**) ∎
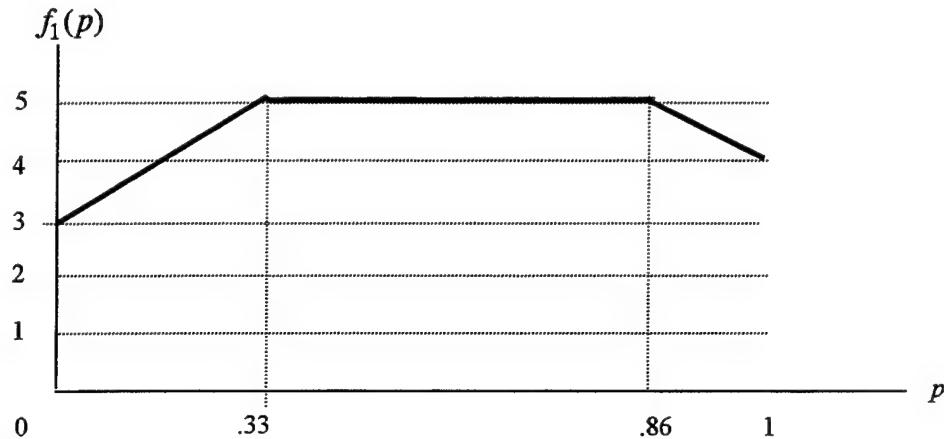
Figure 1.1: The value of the system of **Example 1.1** as a function of
p, as given by formulation [EX1b].

As we expect, $f_1(p)$ in **Example 1.1** is a concave function, which is not easy to minimize. In order to overcome this problem, we reformulate the system-interdiction problem. The idea is to leave the feasible region of the follower independent of **x** and, instead, add a penalty term in the follower's objective function for any use of interdicted activities. The new formulation is convex, for continuous **x**, and a local optimum is a global optimum, too.

We prove in **Chapter III** that there exists a finite penalty multiplier $v*$, such that for all $v \geq v*$ the following is equivalent to formulation [LSIP]:

$$[\text{LSIP-1}] \quad \min_{\mathbf{x} \in X} \quad \max_{\mathbf{y} \in Y} \quad \mathbf{c}^T\mathbf{y} - \mathbf{x}^T V \mathbf{y}$$

$$\text{where} \quad X = \left\{ \mathbf{x} \in \{0,1\}^n \;\middle|\; R\mathbf{x} \leq \mathbf{r} \right\}$$

$$Y = \left\{ \mathbf{y} \mid A\mathbf{y} \leq \mathbf{b}, \; \mathbf{0} \leq \mathbf{y} \leq \mathbf{u} \right\}, \quad \text{and}$$

$$V = \text{diag}(v\mathbf{1}).$$

Let $f_2(\mathbf{x}) \equiv \max_{\mathbf{y} \in Y}\{\mathbf{c}^T\mathbf{y} - \mathbf{x}^T V\mathbf{y}\}$ so that [LSIP-1] is equivalent to $\min_{\mathbf{x} \in X} f_2(\mathbf{x})$. The equivalence of formulations [LSIP] and [LSIP-1] means that $f_2(\mathbf{x}) = f_1(\mathbf{x})$ for all $\mathbf{x} \in X$, and so if $\mathbf{y}(\mathbf{x}) \in \operatorname*{argmax}_{\mathbf{y} \in Y}\{\mathbf{c}^T\mathbf{y} - \mathbf{x}^T V\mathbf{y}\}$, then $\mathbf{x}^T V\mathbf{y} = \mathbf{x}^T\mathbf{y} = 0$ and $f_2(\mathbf{x}) = \mathbf{c}^T\mathbf{y}(\mathbf{x})$ is not actually a function of v.

Notice that when we relax the binary constraints in X, $f_2(\mathbf{x})$ is a convex function, and that the feasible region of the follower's problem is independent of the interdiction plan. More importantly, from basic linear programming theory we can reformulate [LSIP-1] to $\min_{\mathbf{x} \in X}\{z \mid z \geq \mathbf{c}^T\mathbf{y} - \mathbf{x}^T V\mathbf{y} \; \forall \mathbf{y} \in Y'\}$ where Y' is the set of extreme points of $Y = \{\mathbf{y} \mid A\mathbf{y} \leq \mathbf{b}, \; \mathbf{0} \leq \mathbf{y} \leq \mathbf{u}\}$. This suggests solving [LSIP-1] with a row generation algorithm, which is essentially Benders Decomposition (Benders 1962). In this algorithm, the inner minimization is a subproblem that generates extreme points of Y. The master problem is a relaxation of [LSIP-1] that approximates $f(\mathbf{x})$, from below, with cuts constructed from those extreme points. Actually, this algorithm has already been developed and applied to the max-flow network-interdiction problem (Cormican 1995).

Given an interdiction plan $\hat{\mathbf{x}}$ and the solution $\mathbf{y}(\hat{\mathbf{x}})$ of the associated subproblem, the new cut in the master problem, $z \geq \mathbf{c}^T\mathbf{y}(\hat{\mathbf{x}}) - \mathbf{x}^T V\mathbf{y}(\hat{\mathbf{x}})$, will be "reasonably tight," i.e., will give a good approximation of $f_2(\mathbf{x})$ for $\mathbf{x} \in X, \mathbf{x} \neq \hat{\mathbf{x}},$ if we have a valid, but small, penalty multiplier v. For instance, in the max-flow network interdiction case $v = 1$ is always valid and gives tight cuts. However, in another system-interdiction problem, the minimum valid value of v could be difficult to calculate. A "large enough" penalty can be easy to define—$f_2(\mathbf{0})$ will work for some problems—but such a penalty can be much

larger than necessary. On the other hand, if we guess and we guess too low, we might terminate with an incorrect solution and not know it (see **Example 3.1** in **Chapter III**). If we guess and we guess too high, the running time of the decomposition algorithm will be excessive, due to the loose cuts.

The Benders decomposition algorithm is discussed in detail in **Chapter II** (for the MXSP scenario) and in **Chapter III** (for more general system-interdiction problems). We also develop a second decomposition algorithm, which is a variant of the Benders decomposition that assumes no bound on the penalty multiplier v in [LSIP-1], and describe a hybrid decomposition algorithm that is a combination of the first two. The second and third algorithms appear to be superior to the first for MXSP.

Example 1.1 (revisited)

We can reformulate **Example 1.1** in the form of [LSIP-1] and obtain an equivalent (for $p = 0$ and $p = 1$), but convex formulation. We'll do so with two penalties, $v = 2$ and $v = 10$, to show the difficulties that a large penalty may cause. When we relax the binary constraints, the corresponding parametric linear program is:

$$[\text{EX1c}] \quad \min_{0 \le p \le 1} f_2(p)$$

$$\text{where } f_2(p) = \max_{\mathbf{y}} \quad y_1 + y_2 - v\left((1-p)\,y_1 + p\,y_2\right)$$

$$\begin{array}{rrcl}
\text{s.t.} & y_1 & & \le 5 \\
& & y_2 & \le 5 \\
& y_1 - & y_2 & \le 4 \\
& -y_1 + & y_2 & \le 3 \\
& y_1 \ge 0, & y_2 \ge 0. &
\end{array}$$

As can be seen in **Figure 1.2**, for both values of v the solution of the inner

maximization problem, as a function of p, is convex and matches $f_2(p)$ for $p = 0$ and $p = 1$. However, for $v = 10$, the relaxation is much less useful. Assume we start the decomposition algorithm we mentioned earlier, with $p = 0$. The two cuts the subproblem would produce, for $v = 2$ and $v = 10$, are shown in the graph. The cut using $v = 2$ gives a tighter bound on $f_2(p)$ for $p = 1$ and clearly that difference will become more significant in more realistic and multi-dimensional problems. ∎
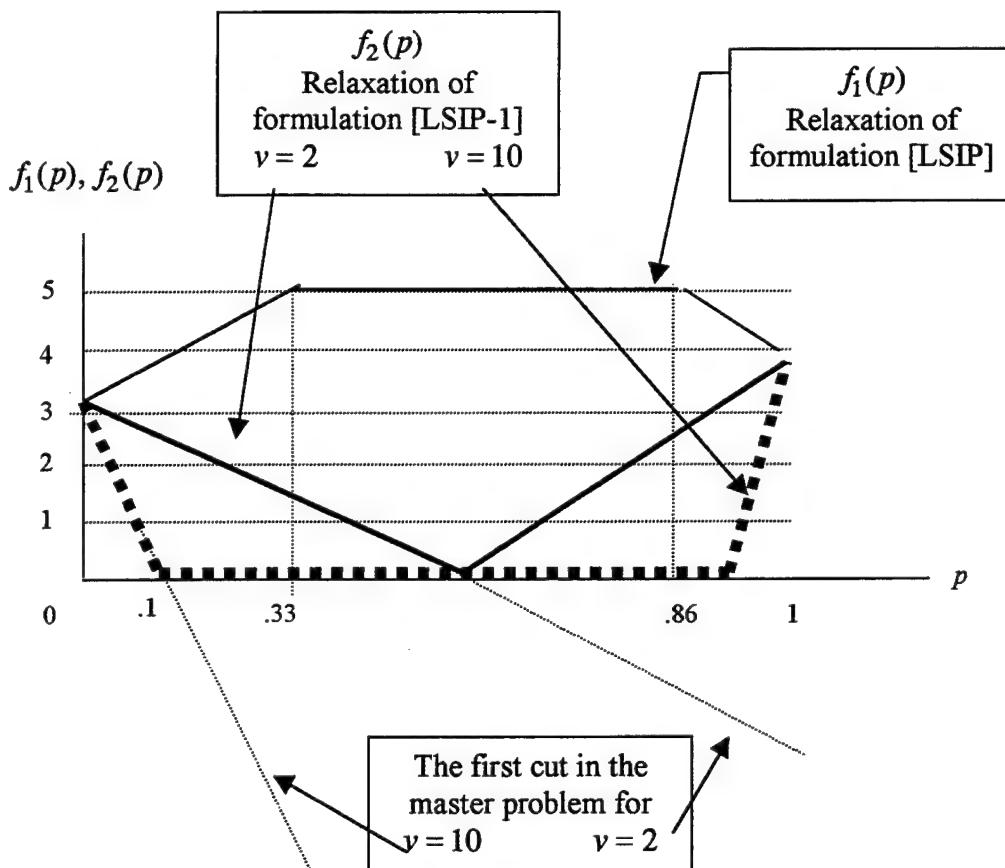
Figure 1.2: Relaxation of different formulations for **Example 1.1**. The relaxation using formulation [LSIP] is concave while the relaxation using formulation [LSIP-1] is convex. When solving formulation [LSIP-1] with Benders decomposition, the cuts in the master problem are tighter for $v = 2$ compared to $v = 10$.

C. SYSTEM DEFENSE

The system-defense problem is a natural extension of the system-interdiction problem. When a system user expects his system to be interdicted, he may wish to expend resources to protect that system to mitigate against the effects of interdiction. The question SD addresses is: How should a system user (now also called the "defender"), with limited resources, "harden" the components of his system to best protect against interdiction, given that the interdictor will optimize his interdiction plan with knowledge of those improvements?

Let the set of feasible defense plans for the system user be given by $G = \{\mathbf{g} \in \{0,1\}^n \mid H\mathbf{g} \le \mathbf{h}\}$ (we assume that G is not empty). We assume absolute protection so that $g_k = 1$ implies that activity j is fully secured against interdiction. SD finds the optimal interdiction strategy for the system user, $\mathbf{g}^*$, and the value of the system associated with this defense plan. Formally, the *Linear System-Defense Problem* (LSDP) is defined as the following problem for the system user:

[LSDP] $\quad \max\limits_{\mathbf{g} \in G} \quad \min\limits_{\mathbf{x} \in X(\mathbf{g})} \quad \max\limits_{\mathbf{y} \in Y(\mathbf{x})} \quad \mathbf{c}^T\mathbf{y}$

where
$$G \quad = \{\mathbf{g} \in \{0,1\}^n \mid H\mathbf{g} \le \mathbf{h}\},$$

$$X(\mathbf{g}) \quad = \left\{\mathbf{x} \in \{0,1\}^n \mid R\mathbf{x} \le \mathbf{r}, \, \mathbf{0} \le \mathbf{x} \le \mathbf{1} - \mathbf{g}\right\}, \quad \text{and}$$

$$Y(\mathbf{x}) \quad = \{\mathbf{y} \mid A\mathbf{y} \le \mathbf{b}, \, \mathbf{0} \le \mathbf{y} \le U(\mathbf{1} - \mathbf{x})\}.$$

[LSDP] can be viewed as a min-max linear system-interdiction problem where the system user is the leader, the interdictor is the follower, and for every feasible defense plan $\mathbf{g}$, the associated value of the system is given by a solution to a system-interdiction

problem where the protected activities are invulnerable (however we assume that $X(\mathbf{g})$ is not empty for all $\mathbf{g} \in G$). Thus, in SD the leader interdicts the interdictor. Notice that a linear program can no longer represent "the system." However, when the defended system is represented by a linear program (as in our formulation of [LSDP]), we can solve the overall system-defense problem through a nested decomposition algorithm, using one or more of the decomposition algorithms developed for [LSIP.]

Example 1.1 (revisited)

Assuming that the system user can protect one of his activities, y_1 or y_2, it is clear that he should protect y_1. However, in more complicated problems, devising the optimal defense plan is much more difficult, and actually the system defense problem is NP-hard and not known to be in NP. ∎

D. OUTLINE

In this introduction, we have defined the system-interdiction problem, which is a difficult-to-solve bi-level mixed-integer programming problem. The continuous relaxation of the first formulation we gave for LSIP is a concave function in the interdiction variables, so even the relaxed problem is difficult to solve. Therefore, we introduced a second formulation of SI that is equivalent to the first for binary interdiction decisions, but is convex when we relax those binary constraints. The second formulation uses a penalty term in the objective function that prevents the follower from using interdicted activities. Unfortunately, this formulation may not be useful if a valid, sufficiently small penalty multiplier, is hard to find.

In **Chapter II** we develop three decomposition algorithms for solving the problem of *Maximizing the Shortest Path* (MXSP) in a directed network. Computational

results for those three algorithms and an MIP formulation of the problem are provided, and practical techniques to speed convergence of the decomposition schemes are described and demonstrated.

In **Chapter III**, we extend the models and methodology of **Chapter II** to solve interdiction problems defined on systems modeled as general linear or integer programs, not just networks. We use:

(a)   A decomposition algorithm that is based on a finite and "good" penalty multiplier,

(b)   A decomposition algorithm that assumes no finite bound on the multiplier, or

(c)   A decomposition algorithm that is a hybrid of first two.

In **Chapter IV**, we discuss the system-defense problem, and solve it using a nested decomposition algorithm. Throughout the Chapter, we use the problem of *Defending the Shortest Path* (DSP) as an illustrative example, but the approach and development is valid for general SD as well. Computational results for DSP are given, and practical techniques to speed convergence of the nested decomposition schemes are demonstrated.

Finally, **Chapter V** extends our approaches further, to one example of "stochastic system interdiction," a case where interdiction successes are uncertain. As expected, this problem is harder to solve and so we suggest several approximation algorithms that are based on the three decomposition algorithms developed earlier. Preliminary computational experience is reported for a stochastic shortest-path network-interdiction problem, but the suggested algorithms can be used for other stochastic system interdiction problems, too.

II. SHORTEST-PATH NETWORK INTERDICTION

Network-interdiction problems involve two opposing forces, a *leader* and a *follower*, who are engaged in a warlike conflict. The follower operates a network so as to optimize his objective function which involves moving a supply convoy through the network as quickly as possible, or maximizing the amount of materiel transported through the network subject to capacity constraints, etc. The interdictor tries to restrict the follower's achievable objective value by interdicting (attacking) arcs so as to destroy those arcs entirely, or increase their effective length, reduce their capacity, etc. The purpose of this chapter is to develop new models and solution methods for the problem of interdicting a road or other transportation network in order to maximize the (post-interdiction) shortest-path length between two specified nodes.

The topic of network interdiction has received some attention over the years, initially with military applications. For instance, McMasters and Mustin (1970) and Ghare *et al.* (1971) develop methods for interdicting a capacitated supply network to hinder the movements of enemy troops and materiel. More recently, research was triggered by drug interdiction efforts (Wood 1993, Washburn and Wood 1994) and by the need to assess the vulnerability of information networks to interdiction (Grötschel *et al.* 1992, Medhi 1994).

The network-interdiction problem we focus on is *Maximizing the Shortest Path* (MXSP) (Fulkerson and Harding 1977, Golden 1978). In this problem, a "network user," i.e., the follower, wishes to traverse a path of minimum length (or minimum time, minimum cost, etc.) between two specified nodes, s and t, in a directed network. But, by first attacking the network using limited resources, an interdictor, i.e., the leader, can

destroy certain arcs, or increase the effective length of certain arcs, and thereby increase that minimum length. MXSP is the interdictor's problem: Subject to a limited interdiction budget (and possibly other restrictions), interdict arcs in a network so as to maximize the shortest-path length between nodes s and t.

In our definition of MXSP, arc interdiction involves a binary decision with known resource consumption and assured success. *The k-most-vital-arcs problem* (Corely and Shaw 1982, Malik *et al.* 1989) is a special case of MXSP where the interdictor seeks to destroy exactly k arcs to interdict the network most effectively. Since that problem is NP-complete (Ball *et al.* 1989), it follows that MXSP is NP-complete.

The k-most-vital-arcs problem has received limited attention and we are not aware of effective algorithms for solving it. Malik *et al.* (1989) suggest a potentially effective algorithm for the problem, but the algorithm has a theoretical flaw as we discuss in Section C. Corely and Shaw (1982) suggest an algorithm for the *single-most-vital-arc problem* but this problem is a very simple case of MXSP which is solvable in polynomial time.

Unlike the k-most-vital-arcs problem, MXSP allows general resource constraints which, most significantly, enable the modeling of different types of interdiction resources, e.g., ground troops, aerial sorties, cruise missiles, etc. Fulkerson and Harding (1977) and Golden (1978) have studied a simpler variant of MXSP incorporating a single type of interdiction resource and arc lengths that increase linearly with the amount of resource applied. We believe that our model with discrete interdiction variables is more realistic.

In this chapter, we first show how to model MXSP as a mixed-integer program (MIP). It is intuitively clear, and later demonstrated by computation, that this MIP can be very difficult to solve directly using LP-based (linear-programming-based) branch and bound. Therefore, we devise three decomposition-based algorithms for MXSP and demonstrate their computational effectiveness. The first algorithm implements a Benders decomposition (Benders 1962) that solves MXSP much like Cormican (1995) solves a maximum-flow *network-interdiction problem*. This technique converges slowly, as does branch and bound applied to the basic MIP, when interdictions cause large local delays. The second decomposition algorithm does not suffer as much from this problem. That decomposition (a) simplifies the master problem of the first algorithm to a set-covering problem (SCP), (b) improves efficiency by incorporating a greedy heuristic for the SCP (in addition to using an exact algorithm), and (c) exploits the special structure of shortest-path problems to gain efficiency. The last algorithm we devise is a hybrid of the first two.

A.   THE BASIC MODEL: MXSP AS A MIXED-INTEGER PROGRAM

The mathematical programming formulation of MXSP on a directed graph $G=(\mathcal{N},\mathcal{A})$ is:

Problem: Maximize the shortest-path length in a directed network by interdicting arcs.

Indices: $i \in N$, nodes in G (s is the source node, t is the sink node),

   $k \in \mathcal{A}$, arcs in G,

   $k \in FS(i)$ ($k \in RS(i)$) arcs directed out of (into) node i,

Data: $0 \leq c_k < \infty$, nominal integer length of arc k,

$0 \leq d_k < \infty$, added integer delay if arc k is interdicted,

$\mathbf{r}$ vector of available interdiction resources,

R matrix of interdiction-to-resource conversions,

Variables: $x_k = 1$ if arc k is interdicted by the leader; else $x_k = 0$,

$y_k = 1$ if arc k is traversed by the follower; else $y_k = 0$,

Formulation:

$$[\text{MXSP-P}] \quad \max_{\mathbf{x} \in X} \min_{\mathbf{y}} \sum_{k \in \mathcal{A}} (c_k + x_k d_k) y_k$$

$$\text{s.t.} \quad \sum_{k \in FS(i)} y_k - \sum_{k \in RS(i)} y_k = \begin{cases} 1 & i = s \\ 0 & i \in N - s - t \\ -1 & i = t \end{cases}$$

$$y_k \geq 0 \quad \forall k \in \mathcal{A}$$

where $X = \left\{ \mathbf{x} \in \{0,1\}^{|\mathcal{A}|} \,\middle|\, R\mathbf{x} \leq \mathbf{r} \right\}$, and:

(a) Node s is the source node and t is the terminal node,

(b) The set of arcs directed out of node i is denoted "$FS(i)$" and the set of arcs directed into node i is denoted "$RS(i)$,"

(c) $x_k = 1$ implies arc k is interdicted; else $x_k = 0$,

(d) Flow-balance constraints (1) in variables $\mathbf{y}$ route one unit of flow from s to t; the inner minimization is a standard shortest-path model with arc lengths $c_k + x_k d_k$,

(e) c_k is the nominal length of arc k and $c_k + d_k$ is the length of that arc if it is interdicted; d_k is finite and comprises such factors as repair time or the length of a local detour (the case of $d_k = \infty$ is dealt with later),

(f) $R\mathbf{x} \leq \mathbf{r}$ is a set of side constraints on interdiction resources; thus, X represents the set of feasible interdiction plans (we assume that X is not empty),

(g)   All data are assumed integral, and $d_k, c_k \geq 0 \ \forall \ k \in A$,

(h)   All solutions will be assumed to be integral since variables $\mathbf{x}$ are required to be integral and extreme points of the inner minimization are well known to be integral.

To simplify presentation, and without loss of generality, we make the further assumption:

Assumption 2.1: *The interdictor has insufficient resources to disconnect s from t.* ■

This assumption is innocuous and merely simplifies presentation of our algorithms. All of the algorithms are easily modified to identify the degenerate case in which s and t can be disconnected. The extensions of our techniques to handle undirected networks and/or node interdiction are also straightforward.

If we fix $\mathbf{x}$, take the dual of the inner minimization in [MXSP-P], make a few simple modifications and then release $\mathbf{x}$, the following MIP results:

$$[\text{MXSP-D}] \quad z^* = \max_{\mathbf{x}, \pi} \quad \pi_t - \pi_s$$

$$\text{s.t.} \quad \pi_j - \pi_i - d_k x_k \leq c_k \quad \forall k \equiv (i,j) \in \mathcal{A}$$

$$\pi_s = 0$$

$$\mathbf{x} \in X$$

Note that $\pi_s = 0$ may be assumed because the inner minimization of MXSP has at least one redundant flow-balance constraint (as do all network flow models containing a balance constraint for each node). Also, note that the dual variables π are unconstrained

in sign and, having reversed their signs compared to the usual convention, we may interpret π_i as the post-interdiction shortest-path distance from s to i.

[MXSP-D] is essentially the model explored by Fulkerson and Harding (1977) and by Golden (1978), except that their variables $\mathbf{x}$ are continuous and only a single resource constraint is allowed. Thus, that model is a simple linear program (LP). Fulkerson and Harding (1977) suggest solving the dual of that LP which may be interpreted as a parametric min-cost flow model. This approach does not appear to be useful when we relax the binary constraints in [MXSP-D] because their model does not allow any additional constraints such as $\mathbf{x} \le \mathbf{1}$.

In theory, we can solve [MXSP-D] using a standard LP-based branch-and-bound algorithm. However, especially when possible delays d_k are large, the LP relaxation of the model is weak and this results in excessive enumeration and unsatisfactory computation times. We use a decomposition approach instead.

B.   A BASIC DECOMPOSITION ALGORITHM

Our basic decomposition algorithm to solve MXSP is a direct application of Benders decomposition to [MXSP-P] (or [MXSP-D] as a MIP, e.g., Garfinkel and Nemhauser 1972, pp. 135-143). Let $\hat{\mathbf{y}} \in \{0,1\}^{|A|}$ denote an arc-path incidence vector corresponding to an s-t path P, i.e., $\hat{y}_k = 1$ implies arc k is in P; otherwise, $\hat{y}_k = 0$. $z(\hat{\mathbf{y}}) = \sum_{k \in A} c_k y_k$ is the length of the path $\hat{\mathbf{y}}$. Let $\hat{Y}$ denote a collection of arc-path incidence vectors corresponding to a subset of all simple s-t paths in G. For simplicity, we refer to $\hat{\mathbf{y}}$ as "a path" and $\hat{Y}$ as "a set of paths." Also, let $D = \text{diag}(\mathbf{d})$ and define:

$$\left[\text{Master}(\hat{Y})\text{-}1\right] \quad z_{\hat{Y}} = \max_{\mathbf{x} \in X} \quad z$$

$$\text{s.t.} \quad z \le \mathbf{c}^T \hat{\mathbf{y}} + \mathbf{x}^T D \hat{\mathbf{y}} \quad \forall\, \hat{\mathbf{y}} \in \hat{Y}$$

$$\left[\text{SP-Sub}(\hat{\mathbf{x}})\right] \quad z_{\hat{\mathbf{x}}} = \min_{\mathbf{y}} \sum_{k \in \mathcal{A}} (c_k + \hat{x}_k d_k) y_k$$

$$\text{s.t.} \quad \sum_{k \in FS(i)} y_k - \sum_{k \in RS(i)} y_k = \begin{cases} 1 & i = s \\ 0 & i \in N - s - t \\ -1 & i = t \end{cases}$$

$$y_k \ge 0 \quad \forall\, k \in \mathcal{A}$$

Let Y denote the set of all simple $s\text{-}t$ paths. For fixed $\mathbf{x} = \hat{\mathbf{x}}$, a solution to the inner minimization of [MXSP-P], which is [SP-Sub($\hat{\mathbf{x}}$)], always occurs at a path $\hat{\mathbf{y}}$. Therefore, [Master($\hat{Y}$)-1] is equivalent to [MXSP-P] when $\hat{Y} = Y$. However, we hope to solve [MXSP-P], at least approximately, by sequentially generating only a small fraction of the extreme points of Y in a decomposition algorithm:

Algorithm 1: Basic Benders decomposition algorithm for MXSP.

Input: An instance of MXSP and allowable optimality gap ε.

Output: Interdiction plan $\mathbf{x}^*$ that solves MXSP to within ε units of optimality.

Step 0: $\hat{Y} \leftarrow \varnothing, \underline{z} \leftarrow -\infty, \bar{z} \leftarrow \infty, \hat{\mathbf{x}} \leftarrow \mathbf{0}$.

Step 1: Solve [SP-Sub($\hat{\mathbf{x}}$)] for solution $\hat{\mathbf{y}}$ with objective $z_{\hat{\mathbf{x}}}$.

$\hat{Y} \leftarrow \hat{Y} \cup \hat{\mathbf{y}}$.

If $\underline{z} < z_{\hat{\mathbf{x}}}$ then $\mathbf{x}' \leftarrow \hat{\mathbf{x}}$ and $\underline{z} \leftarrow z_{\hat{\mathbf{x}}}$.

Step 2: Solve [Master($\hat{Y}$)-1] for solution $\hat{\mathbf{x}}$ with objective $z_{\hat{Y}}$

$\bar{z} \leftarrow z_{\hat{Y}}$.

Step 3: If $\bar{z} - \underline{z} > \varepsilon$ then go to Step 1.

Step 4: $\mathbf{x}^* \leftarrow \mathbf{x}'$, print $\mathbf{x}^*$ and stop.

The correctness of the algorithm, as in any Benders decomposition algorithm, is based on the following observations:

(a) The sub-problem (which is simply a shortest-path problem here) finds an optimal follower's reaction for a specific leader's interdiction plan $\hat{\mathbf{x}}$. Hence, $z_{\hat{\mathbf{x}}}$ gives a lower bound on the leader's optimal solution value. This bound is finite because of **Assumption 2.1**.

(b) If the sub-problem produces the same s-t path twice, the upper bound and the lower bound must match and the algorithm terminates. If the sub-problem continues to find new paths, the algorithm must converge in a finite number of iterations because the number of simple s-t paths is finite. (The number of paths and thus the number of iterations may be exponential, however.)

(c) When $\hat{Y}$ includes all simple s-t paths, [Master(Y)-1] is clearly equivalent to [MXSP-P]. Otherwise, when $\hat{Y} \subseteq Y$, [Master($\hat{Y}$)-1] is a relaxation of [MXSP-P] and thus, $z_{\hat{Y}}$ is an upper bound on the interdictor's optimal objective value. Note: The master problem constraints defined with respect to $\hat{Y}$ are called "Benders cuts."

(d) To tighten the relaxation of [Master($\hat{Y}$)-1], we next introduce "integrality cuts."

Proposition 2.1: *For every Benders cut $z \leq \mathbf{c}^T\hat{\mathbf{y}} + \mathbf{x}^T D\hat{\mathbf{y}}$, the integrality cut $\mathbf{x}^T\hat{\mathbf{y}} \geq 1$ is valid whenever $\underline{z} > \mathbf{c}^T\hat{\mathbf{y}}$.*

Proof: Note that each such Benders cut implies that $z^* \leq \mathbf{c}^T\hat{\mathbf{y}} + \mathbf{x}^{*T}D\hat{\mathbf{y}}$. Furthermore, either $\mathbf{x}^{*T}\hat{\mathbf{y}} = 0$ or $\mathbf{x}^{*T}\hat{\mathbf{y}} \geq 1$. If $\mathbf{x}^{*T}\hat{\mathbf{y}} = 0$ given that $\underline{z} > \mathbf{c}^T\hat{\mathbf{y}}$, then $z^* \leq \mathbf{c}^T\hat{\mathbf{y}} + \mathbf{x}^{*T}D\hat{\mathbf{y}} =$

$\mathbf{c}^T\hat{\mathbf{y}} + 0 < \underline{z}$, which is a contradiction. Therefore, $\mathbf{x}^{*T}\hat{\mathbf{y}} \geq 1$ which implies that $\mathbf{x}^T\hat{\mathbf{y}} \geq 1$ is a valid integrality cut. ∎

Corollary 2.1: For every Benders cut $z \leq \mathbf{c}^T\hat{\mathbf{y}} + \mathbf{x}^T D\hat{\mathbf{y}}$, the integrality cut of **Proposition 2.1**, $\mathbf{x}^T\hat{\mathbf{y}} \geq 1$, can be tightened to $\mathbf{x}^T\hat{\mathbf{y}} \geq 2$ if $\underline{z} > \mathbf{c}^T\hat{\mathbf{y}} + \max_k d_k\hat{y}_k$, it can be tightened to $\mathbf{x}^T\hat{\mathbf{y}}$

≥ 3 if $\underline{z} > \mathbf{c}^T\hat{\mathbf{y}} + \max_{k \neq k'}\{ d_k\,\hat{y}_k + d_{k'}\,\hat{y}_{k'} \}$, and so forth. ∎

Naturally, as $\underline{z}$ is updated in the algorithm, we may be able to tighten previously generated integrality cuts using **Corollary 2.1**, too. Actually, in our implementation, we add the cut $\mathbf{x}^T\hat{\mathbf{y}} \geq 1$ even if $\underline{z} = \mathbf{c}^T\hat{\mathbf{y}}$ (or $\mathbf{x}^T\hat{\mathbf{y}} \geq 2$ even if $\underline{z} = \mathbf{c}^T\hat{\mathbf{y}} + \max_k d_k\hat{y}_k$, etc.). If the optimal solution value exceeds the current value of $\underline{z}$, those cuts are valid. But, if those cuts render the master problem infeasible, the algorithm can be terminated with the incumbent $\mathbf{x}'$ being optimal.

We also improve the effectiveness of **Algorithm 1** by not solving the master problem to optimality. This well-known variant of Benders decomposition (e.g., Geoffrion and Graves 1974) is guaranteed to converge as long as every sub-optimal integer solution $\hat{\mathbf{y}}$ satisfies $\mathbf{c}^T\hat{\mathbf{y}} > \underline{z}$ (recall that data are integral), and we do not update $\bar{z}$ unless the master problem is solved to optimality.

Since all d_k are assumed finite, **Algorithm 1** does not allow the interdictor to completely remove (destroy) an arc. To model the effect of complete arc removal, we can solve the sub-problem with interdicted arcs eliminated, while in the master problem we may be able to define a "sufficiently large" artificial delay (say $|\mathcal{N}|\max_k |c_k|$) on every interdictable arc to keep the Benders cut valid. But, as we shall demonstrate

empirically, the run time of the algorithm grows very quickly with the size of that delay. On the other hand, being too conservative with that artificial delay can lead to difficulties as seen in the following example.

Example 2.1

Consider the network of **Figure 2.1** and an interdictor who can remove any two arcs from that network. One obvious optimal solution to MXSP for this network is to interdict (s,a) and (s,b) so that the shortest s-t path has length 20. Now, let d denote the artificial delay that is to be added to interdicted arcs and suppose that we use **Algorithm 1**, without integrality cuts, to solve this problem.

Initially, the algorithm finds the uninterdicted shortest path s-a-t with length 2. Given that solution for the follower, the leader interdicts (s,a) and (a,t) and the "upper bound" (as calculated in Step 2) is $2 + 2d$, which is valid only if $d \geq 9$. In the next step the follower finds the shortest path after (s,a) and (a,t) are removed from the network. The solution is s-b-t, with length 12. The two Benders cuts in the master problem are:

$$z \leq 2 + dx_{sa} + dx_{at} \qquad \text{from the first iteration, and}$$
$$z \leq 12 + dx_{sb} + dx_{bt} \qquad \text{from the second iteration.}$$

Figure 2.1: Network to illustrate difficulties with artificial delays. Numbers next to arcs are lengths.
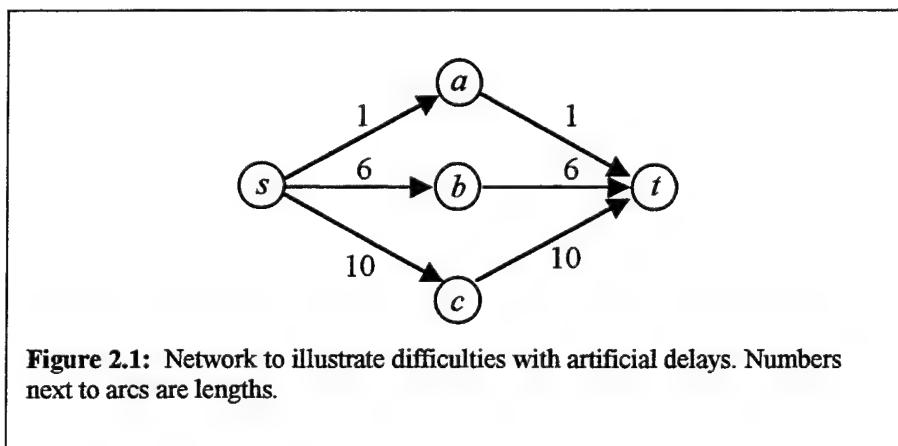
There are four cases to consider now:

(a)   If $d < 5$, the shortest path from the second iteration has length greater the pseudo-upper bound from the first iteration. We conclude then that d is too small, increase it and return to the master problem.

(b)   If $5 \leq d \leq 10$, the master problem objective is 12. Since the lower bound and pseudo-upper bound match, the algorithm terminates, but with an incorrect solution. In this case, we see no way to recognize that d is too small without solving this NP-complete problem: Does there exist a solution to MXSP with objective value greater than $z_{\hat{y}}$?

(c)   If $10 < d < 18$, the master problem interdicts both paths and has optimal objective value $2+d$. The third iteration of the sub-problem finds the path s-c-t with length 20, which is larger than the current pseudo-upper bound. Again, we conclude that d is too small, increase it and return to the master problem.

(d)   If $d \geq 18$, the upper bound is valid and the algorithm terminates with the optimal solution. ∎

C.   A SECOND DECOMPOSITION ALGORITHM

Of course, case (b) of **Example 1** is the most disturbing. To overcome this difficulty, we offer a second decomposition algorithm. This algorithm derives from the variant of Benders decomposition (mentioned earlier) in which the master problem is solved for any feasible solution with objective value greater than the current lower bound. The algorithm iterates until no such solution exists; at that point, the best solution found must be optimal. For simplicity, we now assume that every interdicted arc is completely

removed from the network. No loss of generality arises since arcs with finite delay are easily handled: For any arc k with $d_k < \infty$ and nominal length c_k, create two parallel arcs, k_1 and k_2. Arc k_1 has length c_k and is interdictable, i.e., "removable." Arc k_2 is non-interdictable and has length $c_k + d_k$.

The master problem of the new algorithm simply seeks a feasible solution with objective greater than the current lower bound, $\underline{z} = \max_{\hat{\mathbf{y}} \in \hat{Y}} \mathbf{c}^T \hat{\mathbf{y}}$.

$$\left[\text{Master}(\hat{Y})\text{-}2a\right] \qquad \text{Find} \quad \mathbf{x} \in X$$

$$\text{s.t.} \qquad z \le \mathbf{c}^T \hat{\mathbf{y}} + \mathbf{x}^T D \hat{\mathbf{y}} \quad \forall \hat{\mathbf{y}} \in \hat{Y}$$

$$z \ge \underline{z} + 1$$

(Note that $z \ge \underline{z} + 1$ is sufficient since all data are assumed to be integral.)

Proposition 2.2: *For artificial delay d sufficiently large, **x** is feasible to* [Master($\hat{Y}$)-2a] *if and only if **x** interdicts at least one arc in every path represented by* $\hat{Y}$.

Proof: Sufficiency is guaranteed because $\mathbf{c}^T \hat{\mathbf{y}} \ge 0$ and d is large enough (we can assume $d \ge \underline{z}+1$), and necessity follows because $\underline{z} = \max_{\hat{\mathbf{y}} \in \hat{Y}} \mathbf{c}^T \hat{\mathbf{y}}$. ∎

Instead of solving [Master($\hat{Y}$)-2a], **Proposition 2.2** allows us to solve the following set covering problem (SCP):

$$\left[\text{Master}(\hat{Y}) - 2b\right] \qquad \text{Find} \quad \mathbf{x} \in X$$

$$\text{s.t.} \quad \hat{\mathbf{y}}^T \mathbf{x} \ge 1 \quad \forall \hat{\mathbf{y}} \in \hat{Y}$$

The algorithm we have just established is:

Algorithm 2: A covering decomposition algorithm for MXSP.

Input: An instance of MXSP.

Output: An optimal interdiction plan $\mathbf{x}^*$.

Step 0: $\hat{Y} \leftarrow \emptyset, \underline{z} \leftarrow -\infty, \bar{z} \leftarrow \infty, \hat{\mathbf{x}} \leftarrow \mathbf{0}$.

Step 1: Solve [SP-Sub($\hat{\mathbf{x}}$)] for optimal solution $\hat{\mathbf{y}}$ with objective value $z_{\hat{\mathbf{x}}}$.

$\hat{Y} \leftarrow \hat{Y} \cup \hat{\mathbf{y}}$.

If $\underline{z} < z_{\hat{\mathbf{x}}}$ then $\mathbf{x}' \leftarrow \hat{\mathbf{x}}$ and $\underline{z} \leftarrow z_{\hat{\mathbf{x}}}$.

Step 2: Attempt to solve [Master($\hat{Y}$)-2b] for feasible solution $\hat{\mathbf{x}}$.

If [Master($\hat{Y}$)-2b] is feasible then go to Step 1.

Step 3: $\mathbf{x}^* \leftarrow \mathbf{x}'$, print $\mathbf{x}^*$ and stop.

Let us add a bit of insight to Algorithm 2. Each time the algorithm reaches Step 1, the network user suggests one new s-t path, the best with respect to the interdictor's previous plan. Then, in Step 2, the interdictor tries to find a plan that interdicts all the s-t paths that have been exposed so far, paths represented by $\hat{Y}$. This new interdiction plan may or may not force the network user to traverse a path longer than the current lower bound. Once the interdictor fails to interdict all the paths in $\hat{Y}$, he knows that he cannot force a shortest-path length that is longer than the longest path in $\hat{Y}$. But, this length is exactly the current value of $\underline{z}$, so he concludes that no better interdiction plan than the incumbent $\mathbf{x}'$ exists, and the algorithm terminates.

Algorithm 2 is similar to the algorithm for the k-most-vital-arcs-problem

suggested by Malik *et al.* (1989). There, paths with non-decreasing lengths in the uninterdicted network are enumerated and interdiction is attempted until the l shortest paths cannot be feasibly interdicted. In our setting, this means that an algorithm that produces the lth-shortest path in the original network (e.g., Katoh *et al.*, 1982) replaces Step 1. However, the algorithm in Malik *et al.* assumes that an interdiction plan that interdicts the l shortest paths must correspond to a cutset in the sub-network created from the union of the arcs and nodes from those paths. Consequently, a solution $\mathbf{x}^*$ corresponds to a minimum-cardinality cutset, which can be identified by solving a maximum-flow problem in the sub-network using arc capacities of 1. But, as illustrated next, the assumption is invalid—the master problem just described is a restriction of the correct one—and thus that procedure must be viewed as a heuristic.

Example 2.2

Consider the network of **Figure 2.2**, with all arcs of length 1, and suppose that the interdictor can remove any two arcs from the network. Clearly, the optimal interdiction removes arcs (s,c) and (c,t) and forces a shortest path of length 4. But, deleting two arcs crossing any cutset leaves a shortest path of length 2 or 3. ■

Figure 2.2: Network for Example 2. All arc lengths are 1.
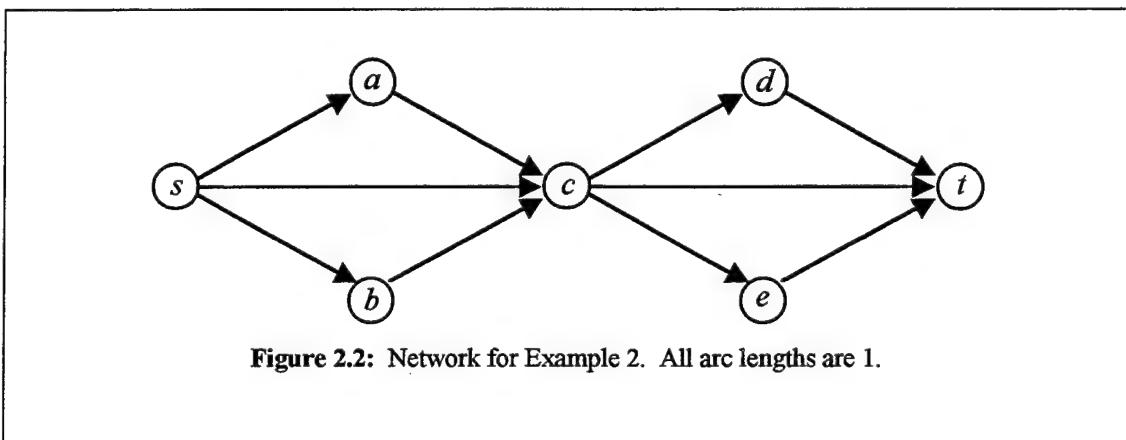
The rest of this section describes several enhancements to **Algorithm 2** to improve efficiency.

When the set X includes only a single resource constraint, say $r^T x \leq r_0$, we can solve the following master problem in place of $[\text{Master}(\hat{Y})\text{-2b}]$:

$$\left[\text{Master}(\hat{Y})\text{-2c}\right] \quad \min_{\mathbf{x}} \quad \mathbf{r}^T \mathbf{x}$$

$$\text{s.t.} \quad \hat{\mathbf{y}}^T \mathbf{x} \geq 1 \quad \forall \hat{\mathbf{y}} \in \hat{Y}$$

$$\mathbf{x} \in \{0,1\}^{|\mathcal{A}|}$$

$[\text{Master}(\hat{Y})\text{-2c}]$ is a standard set-covering problem (SCP) and, if it has an optimal objective value less than or equal to r_0, then $[\text{Master}(\hat{Y})\text{-2b}]$ is feasible. Of course, we need not solve $[\text{Master}(\hat{Y})\text{-2c}]$ to optimality, but just until $\mathbf{r}^T\mathbf{x} \leq r_0$, and this suggests the use of efficient heuristics. We use a version of the simple greedy heuristic for SCPs discussed by, among others, Nemhauser and Wolsey (1988, pp. 466). Many other heuristics exist, (e.g., Beasley 1990, Caprara *et al.* 1996), but this one is easy to implement and provides more-than-adequate performance on our test problems.

Whenever we need to solve a master problem in **Algorithm 2**, we run the greedy heuristic on $[\text{Master}(\hat{Y})\text{-2c}]$. If a feasible solution, $\mathbf{x} \in X$, is found, we proceed to Step 1 of the algorithm. If not, only then do we resort to an exact (and slower) branch-and-bound algorithm.

When solving MXSP, **Algorithm 2** typically iterates much faster than **Algorithm 1** because the master problems are much easier to solve. This is true even when potential delays d_k are small. On the other hand, Algorithm 2 typically requires more iterations than **Algorithm 1**. Another problem with Algorithm 2 is that it incorporates no upper bound, and thus, it cannot be stopped early with a near-optimal solution. To help

overcome these two difficulties, we employ a local-search procedure. With this procedure, we can add more than one path to $\hat{Y}$ per iteration, and we can often determine an upper bound on the optimal solution value.

Let $z(\hat{\mathbf{y}})$ denote the length of path $\hat{\mathbf{y}}$. The key idea behind the local search is that any path $\hat{\mathbf{y}}$ with $z(\hat{\mathbf{y}}) \leq \underline{z}$ may be introduced into $\hat{Y}$ without compromising validity of the algorithm. This is true by definition of the lower bound. There are many ways to find more than one s-t path per iteration and we use the following procedure: It is well known that finding the shortest paths from s to all other nodes is not much more difficult than finding a shortest s-t path, so we first compute the former paths encoded using a standard "shortest path tree" and "predecessor function" (e.g., Ahuja *et al.* 1993, pp.106-107). Let $P(t) = (s, i_1, i_2, \dots, i_n, t)$ be a shortest s-t path and let $P(j)$ be a shortest path to node j. For every node $i_m \in \{i_1, i_2, \dots, i_n\}$, and for every arc (j, i_m) in the network, we build the path $(P(j), i_m, i_{m+1}, \dots, i_n, t)$, represented by its incidence vector $\hat{\mathbf{y}}$, and calculate the path's length $z(\hat{\mathbf{y}})$. Hence, the procedure **Local_Search**(T) takes a shortest path tree T (derived from a shortest-path algorithm) as input and returns a list of paths. We omit pseudo-code for this procedure, and for Procedures **Compare** and **Lift** described below, because their implementations are straightforward given the in-text descriptions.

Every path $\hat{\mathbf{y}}$ returned from **Local_Search** with $z(\hat{\mathbf{y}}) \leq \underline{z}$ is introduced into $\hat{Y}$ as one more path to be covered in the master problem. If $z(\hat{\mathbf{y}}) > \underline{z}$, the path is stored in a special set Y^+. Later, after updating $\underline{z}$ in succeeding iterations, we move any $\hat{\mathbf{y}} \in Y^+$ into $\hat{Y}$ if $z(\hat{\mathbf{y}}) \leq \underline{z}$. Based on Theorem 3 below, the paths contained in Y^+ can also be used to obtain an upper bound $\bar{z}$ on z^* which then allows us to solve for ε-optimal solutions.

Proposition 2.3: *Let $\hat{Y}_{\bar{z}}$ be a set of paths such that $z(\hat{y}) \leq \bar{z}$ for all $\hat{y} \in \hat{Y}_{\bar{z}}$. Then, if* [Master($\hat{Y}_{\bar{z}}$)-2b] *is infeasible, $z^* \leq \bar{z}$.*

Proof: If $z^* > \bar{z}$, we can feasibly interdict all paths $\hat{y}$ such that $z(\hat{y}) \leq \bar{z}$. By assumption we cannot, so $z^* \leq \bar{z}$. ∎

So, if we define the set $\hat{Y}_{\bar{z}} = \hat{Y} \cup \{\hat{y} \in Y^+ \mid z(\hat{y}) \leq \underline{z} + \varepsilon\}$ and [Master($\hat{Y}_{\bar{z}}$)-2b] is infeasible, we know that $z^* \leq \underline{z} + \varepsilon$ and the solution $\hat{x}$ that yielded $\underline{z}$ is ε-optimal.

Our implementation of **Algorithm** 2 uses two additional procedures that empirically speed convergence. The first procedure, **Compare**($\hat{Y}$), returns all the "non-dominated paths" in $\hat{Y}$. Path $\hat{y}_1$ dominates path $\hat{y}_2$ if all interdictable arcs in $\hat{y}_2$ are also contained in $\hat{y}_1$, i.e., if every interdiction plan that interdicts $\hat{y}_1$ also interdicts $\hat{y}_2$. Essentially, **Compare** implements one type of test for "row redundancy" in an SCP. Other redundancy tests are known (e.g., Garfinkel and Nemhauser 1972, pp. 302-304, Taha 1975, pp. 316-332) but this one is easy to implement and has proven to be effective.

The second procedure, **Lift**($\hat{Y}$, $\underline{z}$), uses information about arcs with finite delays to tighten the SCP formulation. Recall that we replace each arc k with length c_k and finite delay d_k with two parallel arcs: Interdictable arc k_1 has length c_k and non-interdictable arc k_2 has length $c_k + d_k$. Now, assume that we have a path $\hat{y}$ in $\hat{Y}$, such that $\hat{y}$ includes arc k_1 and $d_k + z(\hat{y}) \leq \underline{z}$. Then, a path $\tilde{y}$ that is identical to $\hat{y}$ except that it includes arc k_2 instead of k_1 is longer, but is still shorter than the lower bound. Hence, $\tilde{y}$ can be introduced into $\hat{Y}$. Actually, this new path dominates $\hat{y}$ and can replace it, and we have thereby lifted the valid inequality $\hat{y}^T \mathbf{x} \geq 1$ to $\tilde{y}^T \mathbf{x} \geq 1$. (This is, in fact, a "lift" since $\tilde{y} \leq$

$\hat{\mathbf{y}}$, $\hat{y}_{k_1} = 1$ and $\tilde{y}_{k_1} = 0$; see, for example, Nemhauser and Wolsey 1988, pp. 261-267.) The procedure **Lift**$(\hat{Y}, \underline{z})$ returns the set of non-dominated paths generated from $\hat{Y}$ after such replacements. Notice also that if we accept ε-optimal solutions, by **Proposition 2.3** we can introduce $\tilde{\mathbf{y}}$ into $\hat{Y}$ as long as $z(\tilde{\mathbf{y}}) = d_k + z(\hat{\mathbf{y}}) \leq \underline{z} + \varepsilon$. That is done by procedure **Lift**$(\hat{Y}, \underline{z}+\varepsilon)$, which returns the set of non-dominated paths from $\hat{Y}$ with $z(\hat{\mathbf{y}}) \leq \underline{z} + \varepsilon$.

Algorithm 2, with all enhancements is outlined below. The actual implementation reorders certain computations for efficiency's sake.

Algorithm 2E: The covering decomposition of Algorithm 2, enhanced.

Input: An instance of MXSP and optimality tolerance ε.

Output: An ε-optimal interdiction plan $\mathbf{x}^*$ for MXSP.

Step 0: $\hat{Y} \leftarrow \varnothing, Y^+ \leftarrow \varnothing, \underline{z} \leftarrow -\infty, \hat{\mathbf{x}} \leftarrow \mathbf{0}$.

Step 1: Solve [SP-Sub$(\hat{\mathbf{x}})$] for shortest path tree T and objective $z_{\hat{\mathbf{x}}}$. If $\underline{z} < z_{\hat{\mathbf{x}}}$ then $\mathbf{x}' \leftarrow \hat{\mathbf{x}}, \underline{z} \leftarrow z_{\hat{\mathbf{x}}}, \hat{Y} \leftarrow$ **Lift**$(\hat{Y}, \underline{z} + \varepsilon)$.

Step 2: $Y^+ \leftarrow Y^+ \cup$ **Local_Search**(T).

$\hat{Y}' \leftarrow \{\hat{\mathbf{y}} \in Y^+ \mid z(\hat{\mathbf{y}}) \leq \underline{z} + \varepsilon\}, Y^+ \leftarrow Y^+ - \hat{Y}'$.

$\hat{Y}' \leftarrow$ **Lift**$(\hat{Y}', \underline{z} + \varepsilon)$.

$\hat{Y} \leftarrow$ **Compare**$(\hat{Y} \cup \hat{Y}')$.

Step 3: Try to solve [Master$(\hat{Y})$-2b] for optimal solution $\hat{\mathbf{x}}$.

If [Master$(\hat{Y})$-2b] is feasible, then go to Step 1.

Step 4: $\mathbf{x}^* \leftarrow \mathbf{x}'$, print $\mathbf{x}^*$ and stop.

It should be noted that **Local_Search** is also applicable to **Algorithm 1**. In fact, an s-t path derived by any means generates a valid Benders cut for this algorithm. A cut generated from an arbitrary path may be dominated if it contains some non-interdictable arcs, or if the path contains a cycle. Thus, a procedure analogous to **Compare** is also applicable to **Algorithm 1**. (**Lift** does not apply.) In practice, we find that **Algorithm 1** with these modifications requires fewer iterations, but total solution time increases because the master problems quickly become large and hard to solve.

D.   A HYBRID DECOMPOSITION ALGORITHM

The final algorithm we suggest is a hybrid decomposition algorithm that combines the master problems of **Algorithms 1** and **2**. In this hybrid algorithm, Algorithm 3, we view the master problem of **Algorithm 1** as the "basic master problem" and let master problem constraints of Algorithm 2 serve as integrality cuts for the basic master problem. Thus, the master problem of Algorithm 3 integrates the Benders cuts, the integrality cuts of **Algorithm 1** and the covering cuts of **Algorithm 2**. In every iteration we add to the basic master problem one Benders cut, one integrality cut, and we update the set of covering constraints in this master problem using procedures **Local_Search**, **Compare** and **Lift**.

Example 2.3

Consider a network containing s-t paths P_1 and P_2, among others: P_1 traverses arcs 1, 2 and 3 and P_2 traverses arcs 1, 4 and 5. Those arcs have the following parameters: $c_1 = 3$, $c_2 = 1$, $c_3 = 8$, $c_4 = 4$, $c_5 = 6$, $d_1 = 3$, $d_2 = 4$, $d_3 = 5$, $d_4 = 1$, and $d_5 = 3$. Suppose that P_1 is the shortest s-t path in the network, and hence is returned by the sub-problem in the first iteration of the decomposition algorithm. Then, the Benders cut we add to the master

problem is:

$$\text{Benders}_1: \qquad\qquad z \le 12 + 3x_1 + 4x_2 + 5x_3.$$

The integrality cut and the covering constraint associated with this Benders cut are identical:

$$\text{Integrality}_1: \quad x_1 + x_2 + x_3 \ge 1,$$

$$\text{Covering}_1: \quad x_1 + x_2 + x_3 \ge 1,$$

both with "score" 12. (Additional cuts that might be generated by **Local_Search** are ignored.) The score is the uninterdicted length of P_1; this value is important for later tightening or lifting of these cuts. Note that **Compare** will not eliminate one of these constraints, nor would we want it to: A lifted integrality cut is different than a tightened covering constraint even though the base constraints are identical.

Suppose that the interdictor has enough resource(s) to interdict arcs 1, 2 and 3 together, and this is the solution (with $\bar{z}=24$) of the first master problem, which consists of Benders$_1$, Integrality$_1$ and Covering$_1$. Further, assume that the shortest s-t path given these interdictions is P_2 so that $\underline{z} = c_1 + d_1 + c_4 + c_5 = 16$. In this case we can lift the previous covering cut because interdiction of arc 1 alone cannot "push" z over the lower bound $\underline{z}$. (Formally, the difference between the score of Covering$_1$ and $\underline{z}$ exceeds d_1.) The cuts from the first iteration are now:

$$\text{Benders}_1: \qquad\qquad z \le 12 + 3x_1 + 4x_2 + 5x_3,$$

$$\text{Integrality}_1: \quad x_1 + x_2 + x_3 \ge 1, \qquad\qquad \text{Score} = 12,$$

$$\text{Covering}_1: \qquad x_2 + x_3 \ge 1, \qquad\qquad \text{Score} = 15.$$

The score of Covering$_1$ has been updated to 15, which is the length of P_1 with arc 1 interdicted. (Note that we could also have lifted the basic covering cut to $x_1 + x_3 \ge 1$, with

score 16. In our implementation, however, we are satisfied with the first valid lift that we find.)

The cuts from the second iteration are:

Benders$_2$: $z \leq 13 + 3x_1 + 1x_4 + 3x_5$,

Integrality$_2$: $x_1 + x_4 + x_5 \geq 1$,   Score $= 13$,

Covering$_2$: $x_4 + x_5 \geq 1$,   Score $= 16$.

(Again, we ignore covering cuts potentially derived from **Local_Search.**) The score of Integrality$_2$ is the uninterdicted length of P_2 while the score of Covering$_2$ is the length of P_2 with arc 1 (only) interdicted. Now, suppose that the solution to the new master problem, which consists of the two Benders, two integrality and two covering cuts, is $\hat{x}_1 = \hat{x}_3 = \hat{x}_4 = \hat{x}_5 = 1$ and $\hat{x}_2 = 0$. Thus, $\bar{z} = 20$ and the algorithm continues.

Suppose then, at some later iteration, $\underline{z}$ increases to 17. In this case, we can tighten the right-hand side of Integrality$_1$ to 2, because to push z over $\underline{z}$ we must interdict at least two of the three arcs in P_1. (Formally, the difference between $\underline{z}$ and the score of Integrality$_1$ exceeds $\max_{k \in P_1} d_k$.) We can also lift the second covering cut, because interdiction of arc 4 alone cannot push z over $\underline{z} = 17$. The cuts from the first two iterations are now:

Benders$_1$: $z \leq 12 + 3x_1 + 4x_2 + 5x_3$,

Integrality$_1$: $x_1 + x_2 + x_3 \geq 2$,   Score $= 12$,

Covering$_1$: $x_2 + x_3 \geq 1$,   Score $= 15$,

Benders$_2$: $z \leq 16 + 3x_1 + 1x_4 + 3x_5$,

Integrality$_2$: $x_1 + x_4 + x_5 \geq 1$,   Score $= 16$,

Covering$_2$: $x_5 \geq 1$,   Score $= 17$.

The algorithm may or may not halt now depending on the other cuts that have been generated and the value of $\bar{z}$ obtained after solving the current master problem. ∎

In practice, when using a single interdiction resource constraint, we do not use Benders cuts in early iterations. Instead, we heuristically solve the set-covering master problem of Algorithm 2 to suggest a new interdiction plan, as long as this is easy to do. Once the covering problem becomes difficult, or when we want to establish or update an upper bound, we solve the complete, hybrid master problem. If the problem is infeasible, or the value of the objective function (the new value of upper bound) matches the value of the lower bound, we are done. Otherwise, we proceed with the algorithm using the solution from the hybrid master problem.

E. COMPUTATIONAL EXPERIENCE

We use a set of random problems here to test the algorithms we have constructed. Several network structures are used, specified as follows:

(a)   There is one source node s and one sink node t.

(b)   There are $m \times n$ "inner nodes," arranged in a grid of m rows and n columns.

(c)   There is an arc from s to all (inner) nodes in the first column, and there is an arc from all (inner) nodes in the last column to t. None of these $2m$ arcs may be interdicted.

(d)   An arc exists from each node in row r and column c, i.e., in grid position (r,c) to the nodes in positions $(r+1,c)$, $(r-1,c)$, $(r,c+1)$, $(r+1,c+1)$ and $(r-1,c+1)$, assuming that nodes exist in these positions. All of these arcs are interdictable. **Figure 2.3**

gives an example of a test network with $9 = 3 \times 3$ inner nodes.

(e)   The basic data for each network is:

1) m and n,

2)   the identity of interdictable arcs: the total number of potentially interdictable arcs is $a = (n - 2)(5m - 4) + 3m - 2$, but only a specified percentage p of the a arcs are chosen to be interdictable. Interdictable arcs are chosen at random; and

3) r_0, the total interdiction resource available. (We assume single type of interdiction resource.)

Figure 2.3: Example of a 3×3 network for computational tests.
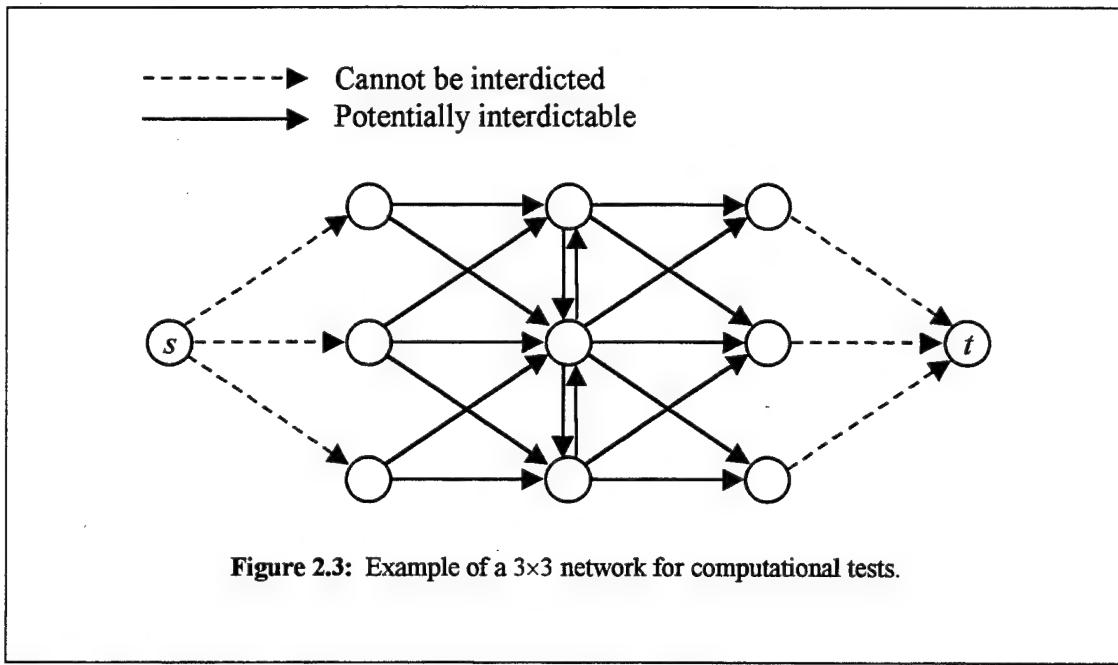
(f)   The randomly generated, integer data for arc k are:

1) c_k, uniformly distributed on $[1,c]$,

2)   when k is identified as interdictable: d_k, uniformly distributed on $[1,d]$, and

3)   when k is identified as interdictable: r_k, uniformly distributed on $[1,r]$.

Our algorithms are programmed in C using the CPLEX version 5.0 callable library (ILOG 1997) for exact solution of master problems, when needed. CPLEX is also used to solve [MXSP-D] directly. Default solver options are used except that "variable selection strategy" is set to "branch based on pseudo reduced cost" when solving the master problem in any of the decomposition algorithms. All computation is performed on an IBM RS-6000 model 590 computer with 512 megabytes of RAM. All running times displayed are averages across ten networks of identical topology, but with different random arc attributes.

Note that in **Algorithms 1** and **3**, we do not solve the master problem to optimality, but rather for a feasible integer solution $\hat{x}$ that yields $z > \underline{z}$. Experience indicates that, when the master problem becomes difficult, it is best to stop with the first such incumbent solution. On the other hand, the first incumbent does not usually generate a "good" cut in early iterations. Our implementation exploits this experience using a simple rule: If we have not solved the master problem to optimality in three seconds, we stop if the incumbent has $z > \underline{z}$, or else we continue until we find such an incumbent or until the master problem is proven infeasible.

Table 1 shows results for problems 1-4. Overall, **Algorithm 3** has the best running times and can be 40 times faster than solving [MXSP-D] directly by branch and bound. **Algorithm 2** is fastest for smaller instances (but without upper bound information during execution). **Algorithm 1** is the slowest of the four procedures. We would like to emphasize a few points:

(a)   Varying arc attributes while holding the network topology and algorithm fixed can lead to widely varying solution times: Compare means and standard

deviations for the running times. In the larger networks, the fastest run (among the 10 different runs) may be 100 times faster than the slowest. We are still investigating ways to reduce the running times of the longer-running problems.

		[MXSP-D]		Algorithm 1				Algorithm 2E				Algorithm 3			
Problem	r_0	T_0	S_0	T_1	S_1	N_1	P_1	T_2	S_2	N_2	P_2	T_3	S_3	N_3	P_3
1	20	107	77	110	115	51	102	2	1	21	320	2	1	20	315
2	30	978	1215	(6)	–	–	–	25	18	36	690	33	36	36	630
3	40	(7)	–	–	–	–	–	650	560	57	1205	220	185	51	1220
4	50	–	–	–	–	–	–	(5)	–	–	–	(7)	–	–	–

Table 2.1: Computational results for a network with $100=10\times10$ inner nodes ($a=396$), $p=100\%$, $c=10$, $d=10$ and $r=5$. The numbers in parentheses represent the number of problems solved to optimality, out of 10 cases, within 3600 CPU seconds.

Legend: T_h –Run time in CPU seconds for Algorithm h.
  (0 = branch-and-bound on [MXSP-D], 2=2E)
  S_h – Standard deviation in CPU seconds of T_h
  N_h – Number of iterations for Algorithm h.
  P_h – Number of constraints in the master problem when the algorithm h terminates.

(b) All the algorithms are very sensitive to r_0, the total available interdiction resource. Running time typically increases rapidly as r_0 increases from a small value but then starts decreasing for sufficiently large values, beyond those displayed here. (Variations in run times occur with changes in other data, but the basic trend remains.) This makes sense since increasing interdiction resource allows more combinations of arcs to be interdicted, up to a point, but then for sufficiently large r_0, all or nearly all arcs can be interdicted, and the problem becomes relatively easy.

(c) In all problem instances, all of the algorithms find good solutions quickly: Most of the running time is spent proving, or trying to prove, optimality.

(d) **Table 2.2** displays results from runs designed to explore the sensitivity of the algorithms to network shape. The decomposition algorithms prefer "tall networks," like the 12×8 network, over "long networks" like the 7×14 network. This tendency may result from the greater number of paths in a long network, the potentially greater number of constraints in the corresponding master problems, and because there is a positive correlation between the number of potential constraints and the actual number needed to generate a tight master problem. However, when not all of the arcs are interdictable (problems 8 and 9), the decomposition algorithms handle long networks quite well (perhaps because there are fewer potential constraints in the master problem). Branch-and-bound for [MXSP-D] seems to perform better on long networks.

				[MXSP-D]		Algorithm 2E				Algorithm 3			
Problem	$m \times n$	a	p	T_0	S_0	T_2	S_2	N_2	P_2	T_3	S_3	N_3	P_3
5	12×8	370	100%	415	665	1	1	18	195	1	1	18	240
6	8×12	382	100%	350	375	140	130	45	1075	100	70	43	1125
7	7×14	405	100%	182	210	(8)	–	–		(8)	–	–	–
8	10×20	876	50%	98	135	30	51	40	495	58	83	38	535
9	10×40	1796	25%	85	140	20	24	55	400	62	90	52	480

Table 2.2: Computational results for networks with different network shapes, with r_0=25, c=10, d=10 and r=5. The total number of potentially interdictable arcs is a and the percentage of interdictable arcs from a is p. See Table 1 for other definitions.

It may be possible to improve computation times substantially by settling for a slightly less-than-optimal solution. So, we next repeat the tests on particularly difficult problems, problems 3, 4 and 7, but allow a 5% optimality gap. Results are displayed in **Table 2.3**. (Optimality gaps were described previously in absolute terms. Here, an allowable gap of $g\%$ means $100(\bar{z}-\underline{z})/\underline{z} \le g$.) Indeed, run times can be significantly shortened.

	[MXSP-D]		Algorithm 2E				Algorithm 3			
Problem	T_0	S_0	T_2	S_2	N_2	P_2	T_3	S_3	N_3	P_3
3	850	810	233	183	43	1070	114	77	40	1075
4	(7)	–	(7)	–	–	–	960	815	56	1620
7	48	37	(9)	–	–	–	112	127	41	1770

Table 2.3: Computational tests on problems from Tables 1 and 2 with a 5% optimality gap allowed. See Table 1's legend for definitions.

In **Table 2.4** we compare the algorithms for the case in which an interdicted arc is actually removed from the network. We fix the cost of interdiction to one unit of resource per arc, so we are actually solving the k-most-vital-arcs problem for k=5 and k=10 in 7×7, 10×10 and 14×14 networks. Note that for standard branch and bound solving [MXSP-D] and for **Algorithm 1**, we use artificial delays of d=5 and d=10. However, d=5 is often too small and yields incorrect solutions while d=10 results in long run times. Branch and bound is the slowest algorithm on these problems, and Algorithm 2 is the fastest by a substantial margin. Results for Algorithm 3 are omitted since that algorithm is slower than Algorithm 2. (We can view Algorithm 3 as Algorithm 2 with Benders cuts added in the master problem. But, these cuts are weak for large d, do not

add much information to Algorithm 2's master problem and mostly serve to hinder solutions.) The table demonstrates the sensitivity of run times to network size and k.

Problem	k	$m \times n$	a	[MXSP-D]				Algorithm 1				Alg. 2E
				$d=5$		$d=10$		$d=5$		$d=10$		
8	5	7×7	188	2.6	[7]	20.3		1.3	[7]	2.8		0.2
9	10	7×7	188	70.4	[4]	(5)		46.6	[5]	180.1		3.5
10	5	10×10	396	28.0	[6]	155.6		4.9	[6]	14.9		0.4
11	10	10×10	396	1334.0	[5]	(0)		137.8	[6]	960.2		21.3
12	5	14×14	860	103.7	[9]	1353.0		14.6	[9]	43.7		1.7

Table 2.4: Results for the k-most-vital-arcs problems. Numbers in parentheses are the number of problems solved, in 10 trials, within 3600 CPU seconds (each). In the columns under $d=5$, numbers in brackets are the number of problem solved correctly, out of the 10 trials. Numbers not in parentheses or brackets in the "algorithm columns" are CPU seconds averaged over 10 trials.

Legend: k – Number of arcs the interdictor may interdict.
(Every arc is potentially interdictable.)
d – Artificial delay. (Other data as in Tables 1-3.)

F.   CONCLUSIONS

This chapter has discussed a shortest-path network-interdiction problem, MXSP, on a directed network. The objective of "the interdictor" is to attack (interdict) network arcs, using limited resources, so as to maximize the length of a shortest path between two specified nodes. Interdiction of an arc increases its effective length, or destroys the arc making it impassable. The ultimate purpose of the interdiction is to slow the movement of the "network user" through a road or other transportation network.

MXSP is an NP-complete, max-min problem. We show how to formulate the problem as a mixed-integer program (MIP) but develop decomposition techniques that typically solve test problems much more efficiently than does LP-based branch and bound with the MIP. Our first technique applies Benders decomposition with a standard master problem and shortest-path sub-problems, but the second decomposition uses a unique set-covering master problem. A third decomposition algorithm is a hybrid of the first two. Special techniques, including integrality cuts for the master problem and local search to generate more than one Benders cut per iteration, significantly improve efficiency over naïve implementations of the decompositions. Numerous avenues are open for further research. These are discussed in **Chapter VI**, Conclusions.

It is clear that our techniques may be generalized to "system interdiction problems," as we shall demonstrate in next chapter. Later, we use these generalizations to solve a "system-defense problem," in particular, the problem of hardening a road network against attack; see **Chapter IV**. The issue of uncertainty in interdiction success for MXSP is discussed in **Chapter V**.

III. THE SYSTEM-INTERDICTION PROBLEM

The mathematical study of interdiction has, until now, focused on "network interdiction" in which an enemy's supply lines, modeled as a network, are efficiently disrupted by attacking network components, e.g., bridges, roads, rail lines, etc. The purpose of this chapter is to generalize the network-interdiction techniques of **Chapter II** to handle the interdiction of general systems, for instance, a segment of an economy that is producing war materiel.

Our basic system-interdiction model assumes that the interdictor makes resource-constrained, binary interdiction decisions to attack a system whose optimal operation is modeled through a mixed-integer linear program. We suggest solving this model using extensions of the three decomposition algorithms developed in **Chapter II**. We then extend those three decomposition algorithms even further, to solve a more general system-interdiction problem, where the optimal system operation is modeled through an even more general optimization problem.

A. WHEN SYSTEM OPERATION CAN BE MODELED WITH A MIXED INTEGER (LINEAR) PROGRAM

In this section, we assume that the optimal solution of the follower's system can be adequately modeled through the optimization of an MIP. Let $U = \text{diag}(\mathbf{u})$. Then, the *Mixed-Integer Linear System-Interdiction Problem* (MILSIP) is defined to be the following leader's problem:

[MILSIP] $\qquad z^* = \min_{\mathbf{x} \in X} \ f_1(\mathbf{x}) \qquad$ where $f_1(\mathbf{x})$ is defined by

[M-Sub(x)-1] $\qquad f_1(\mathbf{x}) = \max_{\mathbf{y} \in Y(\mathbf{x})} \ \mathbf{c}^T \mathbf{y}$

$$\text{and} \qquad X \quad = \left\{ \mathbf{x} \in \{0,1\}^n \ \middle| \ R\mathbf{x} \leq \mathbf{r} \right\},$$

$$Y(\mathbf{x}) \ = \left\{ \mathbf{y} \ \middle| \ A\mathbf{y} \leq \mathbf{b}, \ \mathbf{0} \leq \mathbf{y} \leq U(1-\mathbf{x}), \mathbf{y} \in Y_{INT} \right\},$$

where $\mathbf{c}, \mathbf{y}, \mathbf{u} \in \mathfrak{R}^n$, $\mathbf{c}, \mathbf{b} \in \mathfrak{R}^m$, $A \in \mathfrak{R}^{m \times n}$ and Y_{INT} represents integer (or binary) restrictions on none, some or all of the variables $\mathbf{y}$. With the exception of the set Y_{INT}, formulation [MILSIP] is equivalent to formulation [LSIP], described in **Chapter I**. Thus, $x_j = 1$ means that activity j is interdicted, and that changes the upper bound on y_j from u_j to 0. For notational simplicity, this model assumes that every activity is potentially interdictable but, in practice, certain activities will be off-limits, inaccessible or otherwise unavailable for interdiction. A more significant assumption for modeling purposes is:

Assumption 3.1: The set X is not empty and the inner maximization is feasible for every interdiction plan $\mathbf{x}$. ■

One can imagine more complicated problems where the interdictor's actions affect more than one activity at a time, or where those actions change the costs of the follower's activities or his available resources. The following proposition shows that [MILSIP] can be modified to handle such situations.

Proposition 3.1: *Let*

[MILSIP-1] $\min\limits_{\hat{\mathbf{x}} \in \hat{X}} \max\limits_{\hat{\mathbf{y}} \in Y(\hat{\mathbf{x}})} \hat{\mathbf{c}}^T \hat{\mathbf{y}} - \hat{\mathbf{x}}^T \hat{V} \hat{\mathbf{y}}$

where $\hat{X} = \left\{ \hat{\mathbf{x}} \in \{0,1\}^{\hat{n}} \,\Big|\, \hat{R}\hat{\mathbf{x}} \leq \hat{\mathbf{r}} \right\}$, and

$Y(\hat{\mathbf{x}}) = \left\{ \hat{\mathbf{y}} \in \Re^{\hat{m}} \,\Big|\, \hat{A}\hat{\mathbf{y}} \leq \hat{\mathbf{b}} - \hat{B}\hat{\mathbf{x}}, 0 \leq \hat{\mathbf{y}}, \hat{\mathbf{y}} \in \hat{S} \right\}$

where $\hat{B}_{ij} \geq 0 \;\; \forall i,j$. *Then,* [MILSIP-1] *can be transformed into formulation* [MILSIP].

Proof: See **Appendix B**. ∎

Remark: The restriction $\hat{B}_{ij} \geq 0$ is acceptable, because we don't expect that an interdiction would relax any of the system's constraints.

   We would like to solve [MILSIP] with **Algorithm 1**, the Benders decomposition, but in order to do so we need to reformulate the problem. In Benders decomposition the feasible region of the subproblem is fixed, independent of the first level variables (**x** in our case,) while the objective function changes at every iteration. To obtain this situation in our case, we force the interdiction through a penalty term in the objective function, which will ensure that the use of an interdicted activity is not cost-effective. Then, we can leave interdicted activities free in the subproblem (their upper bounds are not affected by **x**), knowing for sure that these activities will not be used in an optimal solution. In some problems like the *max-flow network-interdiction problem*, however, an "exact penalty" of 1 allows an interdicted activity to be used without compromising equivalence of the models, at least in terms of **x** (Cormican *et al.* 1997). The following proposition gives us a more general result:

Proposition 3.2: *When* **Assumption 3.1** *holds, there exists $v^* < \infty$ such that, for every $v \geq$*

v^*, formulation [MILSIP] and the following problem, with $V \equiv \mathrm{diag}(v\mathbf{1})$, have the same

set of optimal solutions in **x** *and* **y**:

[MILSIP - 2] $z^{**} = \min_{\mathbf{x} \in X} \; f_2(\mathbf{x})$ *where $f_2(\mathbf{x})$ is defined by*

[M - Sub(x) - 2] $f_2(\mathbf{x}) = \max_{\mathbf{y} \in Y} \; \mathbf{c}^T \mathbf{y} - \mathbf{x}^T V \mathbf{y}$

and $X = \left\{ \mathbf{x} \in \{0,1\}^n \; \big| \; R\mathbf{x} \leq \mathbf{r} \right\},$

     $Y = \left\{ \mathbf{y} \; \big| \; A\mathbf{y} \leq \mathbf{b}, \; \mathbf{0} \leq \mathbf{y} \leq U, \mathbf{y} \in Y_{INT} \right\}.$

Proof: First we show that for any $\mathbf{x} \in X$ there exists $v(\mathbf{x})$ such that for every $v \geq v(\mathbf{x})$,

$f_1(\mathbf{x}) = f_2(\mathbf{x})$, and $\arg \max_{\mathbf{y} \in Y(\mathbf{x})} f_1(\mathbf{x}) = \arg \max_{\mathbf{y} \in Y} f_2(\mathbf{x})$. To do so, it suffices to show that

(a)   Every optimal solution of [M-Sub(x)-1] is feasible to [M-Sub(x)-2] with equal

     objective function value—that is trivial—and,

(b)   Every optimal solution of [M-Sub(x)-2] is feasible to [M-Sub(x)-1] with equal

     objective function value. To show that we, need to find $v(\mathbf{x})$ such that for every v

     $\geq v(\mathbf{x})$ every optimal solution **y** of [M-Sub(x)-2] satisfies $\mathbf{x}^T V \mathbf{y} = \mathbf{x}^T \mathbf{y} = 0$.

     To show (b), define $\delta_{\mathbf{x}} = \min \left\{ \mathbf{x}^T \mathbf{y} \; \big| \; \mathbf{y} \in Y, \mathbf{x}^T \mathbf{y} > 0 \right\}$, $M_{\mathbf{x}} = \max_{\mathbf{y} \in Y} \mathbf{c}^T \mathbf{y} - f_1(\mathbf{x})$

(note that $f_1(\mathbf{x})$ is clearly bounded) and finally $v_{\mathbf{x}} = 1 + M_{\mathbf{x}}/\delta_{\mathbf{x}}$. Then, if the optimal **y**

is such that $\mathbf{x}^T \mathbf{y} > 0$, $f_2(\mathbf{x}) \leq \max_{\mathbf{y} \in Y} \mathbf{c}^T \mathbf{y} - \delta_{\mathbf{x}} v_{\mathbf{x}}$ which is a contradiction because, by

definition, $\max_{\mathbf{y} \in Y} \mathbf{c}^T \mathbf{y} - \delta_{\mathbf{x}} v_{\mathbf{x}} < f_1(\mathbf{x})$ and from (a) we know that $f_1(\mathbf{x}) \leq f_2(\mathbf{x})$.

The number of feasible solutions $\mathbf{x} \in X$ is finite. So, let $v^* = \max_{\mathbf{x}} \{v(\mathbf{x}) \mid \mathbf{x} \in X\}$. ∎

Corollary 3.1: MILSIP has an optimal solution where the follower's part of the solution is a vertex of $Y = \{\, \mathbf{y} \mid A\mathbf{y} \geq \mathbf{b}, \mathbf{0} \leq \mathbf{y} \leq \mathbf{u}, \mathbf{y} \in INT_Y\}$. ∎

Remark: Proposition 3.2 is a variation on Morton and Wood (1999). It is shown there that $f_1(\mathbf{x}) = f_2(\mathbf{x})$ for all $\mathbf{x} \in X$, when $\mathbf{y}$ is continuous, $\overline{\pi}_j$ is an upper bound on the optimal dual multiplier for the constraint $y_j \leq u_j(1 - x_j)$ in [M-sub(x)-1] taken over all $\mathbf{x} \in X$ and $V = \mathrm{diag}(\overline{\pi})$. (Note that computing the best possible bound may require full enumeration of the system value for every possible interdiction plan.) This approach does not work when we allow discrete variables $\mathbf{y}$.

Let $V \equiv \mathrm{diag}(v^*\mathbf{1})$. Based on **Proposition 3.2** and **Corollary 3.1**, we can use **Algorithm 1**, the Benders decomposition, with formulation [MILSIP-2]. Define:

$$[\text{Master}(\hat{Y})] \quad \min_{\mathbf{x} \in X, z} \quad z$$

$$s.t. \quad z \geq \mathbf{c}^T \hat{\mathbf{y}} - \mathbf{x}^T V \hat{\mathbf{y}} \qquad \forall \hat{\mathbf{y}} \in \hat{Y}$$

and apply **Algorithm 1** using this master problem and the subproblem [M-Sub(x)-2]. Furthermore, it is clear that the integrality cuts of **Chapter II** are valid here, too, and can tighten the relaxation of $[\text{Master}(\hat{Y})]$ (see **Proposition 2.1** and **Corollary 2.1**). (Straightforward adjustments are required since MXSP is a max-min interdiction problem, while the development in this chapter is for min-max system interdiction.)

As discussed in **Chapter II** with respect to MXSP, the subproblem of **Algorithm 1** finds an optimal reaction of the follower to a specific interdiction plan, which is feasible to the leader. Hence, $f_2(\mathbf{x})$ gives a lower bound on the leader's optimal solution value. The master problem includes only a subset of the follower's vertices and hence it yields an upper bound for the leader. Moreover, the feasible region of the subproblems is fixed, with a finite number of vertices, and in every iteration of the subproblem, there is a solution that is a different vertex of Y. Thus, the algorithm must converge.

We can sometimes refine the penalty term to tighten the master problem. For instance, the penalty for different activities may be different. Such a modification is essentially what we have in MXSP, where the penalty matrix $V \equiv D$ represents the local delays on each arc. Moreover, given an interdiction plan $\hat{\mathbf{x}}$, the penalty multipliers $v_j(\hat{\mathbf{y}})$ for each activity j, can be functions of the optimal solution of the subproblem $\hat{\mathbf{y}} \equiv \hat{\mathbf{y}}(\hat{\mathbf{x}})$, as long as the cut we add to the master problem, $z \geq \sum_j c_j \hat{y}_j - \sum_j x_j v_j(\hat{\mathbf{y}}) \hat{y}_j$, is valid for all $\mathbf{x} \in X$. (But, V cannot be a function of $\mathbf{x}$ since the constraints in the master problem would then be nonlinear.)

V could also have non-zero, off-diagonal entries representing second-order effects. For instance, suppose that the two components of the system under study act "serially" so that destroying either one is as good as destroying both, i.e., $y_1 = y_2$. If that Benders cut

$$z \leq \mathbf{c}^T \mathbf{y} + \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} v & 0 \\ 0 & v \end{bmatrix}$$

is valid, then so is the tighter cut

$$z \le \mathbf{c}^T \mathbf{y} + [x_1 \; x_2] \begin{bmatrix} v & -.5v \\ -.5v & v \end{bmatrix}.$$

Unfortunately, **Proposition 3.2** and the discussion afterward do not suggest a general technique to determine a valid and effective penalty matrix V. Sometimes the structure of the system suggests one, as in MXSP with finite delays, but that is a special case. Recall **Example 2.1**, where we show the difficulties that an insufficiently large penalty might cause. On the other hand, the running time of **Algorithm 1** can be excessive if we use a large penalty (compare results for $d = 5$ and $d = 10$ in **Table 3.4**). Therefore, we wish to devise an algorithm for MILSIP, similar to **Algorithm 2E**, that does not assume any bound on the local effect of an interdiction.

Following the arguments upon which **Algorithm 2** is based, we assume large penalty multipliers and wish to solve [Master($\hat{Y}$)] for the first feasible solution with objective value greater the current lower bound. This is accomplished by solving the following set-covering problem (SCP):

[Master($\hat{Y}$) - 1]   Find $\mathbf{x} \in X$

                      s.t. $\mathbf{I}(\hat{\mathbf{y}})^T \mathbf{x} \ge 1$ $\forall \, \hat{\mathbf{y}} \in \hat{Y}$

where $I_j(\hat{\mathbf{y}}) = 1$ if $\hat{\mathbf{y}}_j > 0$ and $I_j(\hat{\mathbf{y}}) = 0$ if $\hat{\mathbf{y}}_j = 0$. We can do so because every $\mathbf{x}$ feasible in [Master($\hat{Y}$)-1] interdicts one of the basic (and positive) variables in every vertex in $\hat{Y}$, so $\mathbf{x}$ must be feasible in [Master($\hat{Y}$)] and the leader's objective there exceeds the lower bound, for v sufficiently large. In [Master($\hat{Y}$)-1], the interdictor tries to interdict all the vertices thus far exposed by the follower, so "vertices" take the place of "paths" in the discussion of **Algorithm 2** in **Chapter II**.

Analogs of the various enhancements we have suggested for solving MXSP with **Algorithm 2** are applicable to solving MILSIP, too. To begin with, a local-search procedure can be used to generate more than one vertex of the follower's feasible region per iteration. For instance:

(a)   If a linear program represents the follower's system, we can use the last simplex tableau to reach some or all of the neighboring extreme points to the optimal solution. (Reaching all neighboring extreme points could require too much work, but a fixed computational budget could be allocated for finding some subset of these points.)

(b)   If MIP represents the follower's system and [M-Sub($\mathbf{x}$)-1] is solved by branch and bound, feasible solutions found during the enumeration could be used in place of a local-search procedure.

Procedure **Lift** in **Algorithm 2** is based on the penalties d_k used in **Algorithm 1**, and it must be modified to accompany the flexibility in the penalty matrix for the more general case. (If v is "very large," it is likely that **Lift** will have no effect.) Similarly, procedure **Compare** should be modified if the penalty matrix has non-zero off-diagonal elements. However, when we use the same diagonal penalty matrix V for all the solutions of the subproblem, procedures **Compare** and **Lift** remain as they were in MXSP.

Lastly, we can solve MILSIP with a hybrid decomposition algorithm, just as we use **Algorithm 3** for MXSP.

B. INTERDICTION OF EVEN MORE GENERAL SYSTEMS

We now generalize our results for interdiction of general systems, where optimal solution of the follower's system can be adequately modeled through an arbitrary optimization model. Thus, the *General System Interdiction Problem* (GSIP) is defined to be the following leader's problem:

$$[\text{GSIP}] \qquad z^* = \min_{\mathbf{x} \in X} f(\mathbf{x}) \qquad \text{where } f(\mathbf{x}) \text{ is defined by}$$

$$[\text{G - Sub}(\mathbf{x})] \qquad f(\mathbf{x}) = \max_{\mathbf{y} \in Y(\mathbf{x})} g(\mathbf{x},\mathbf{y})$$

and let $\hat{\mathbf{y}}(\hat{\mathbf{x}}) = \operatorname*{argmax}_{\mathbf{y} \in Y(\mathbf{x})} g(\hat{\mathbf{x}},\mathbf{y})$. (We assume that $Y(\mathbf{x})$ is non-empty, $g(\mathbf{x},\mathbf{y})$ is bounded over $Y(\mathbf{x})$ for all $\mathbf{x} \in X$ and the argmax is always unique.)

For constructing a Benders-type decomposition for solving GSIP, essentially **Algorithm 1**, we reformulate the problem as follows:

Proposition 3.3: *Assume that for every* $\hat{\mathbf{x}} \in X$, *we can find a scalar* $c(\hat{\mathbf{x}})$ *and vector of penalty multipliers* $\mathbf{v}(\hat{\mathbf{x}})$ *that satisfy*

$$f(\hat{\mathbf{x}}) = c(\hat{\mathbf{x}}) - \mathbf{v}(\hat{\mathbf{x}})^T \hat{\mathbf{x}} \qquad\qquad [3.3.1], \quad and$$

$$f(\mathbf{x}) \geq c(\hat{\mathbf{x}}) - \mathbf{v}(\hat{\mathbf{x}})^T \mathbf{x} \qquad \forall\, \mathbf{x} \in X. \qquad [3.3.2].$$

Then, $z^* = z^{**}$ *where* z^* *is defined by* [GSIP], *and* z^{**} *is defined by*

$$[\text{GSIP - 1}] \qquad z^{**} = \min_{z,\mathbf{x}} z$$

$$\text{s.t.} \quad z \geq c(\hat{\mathbf{x}}) - \mathbf{v}(\hat{\mathbf{x}})^T \mathbf{x} \qquad \forall\, \hat{\mathbf{x}} \in X.$$

Furthermore, [GSIP] *and* [GSIP-1] *share the same set of optimal solutions in* **x**.

Proof: From formulation [GSIP-1], $z^{**} = \min_{\mathbf{x} \in X} z(\mathbf{x})$ where $z(\mathbf{x}) = \max_{\hat{\mathbf{x}} \in X} \{c(\hat{\mathbf{x}}) - \mathbf{v}(\hat{\mathbf{x}})^T \mathbf{x}\}$, and conditions [3.3.1] and [3.3.2] ensure that $z(\mathbf{x}) = f(\mathbf{x})$ for all $\mathbf{x} \in X$. ∎

Corollary 3.2: For every $\hat{\mathbf{x}} \in X$ such that $\mathbf{v}(\hat{\mathbf{x}})^T \hat{\mathbf{x}} = 0$, $c(\hat{\mathbf{x}}) \equiv f(\hat{\mathbf{x}})$ is necessary to satisfy condition [3.3.1]. ∎

A straightforward implementation of Benders decomposition to formulation [GSIP-1] defines:

$$\left[\text{Master}(\hat{X})\right] \quad \min_{z,\mathbf{x}} \quad z$$

$$\text{s.t.} \quad z \geq c(\hat{\mathbf{x}}) - \mathbf{v}(\hat{\mathbf{x}})^T \mathbf{x} \qquad \forall\, \hat{\mathbf{x}} \in \hat{X},$$

where $\hat{X}$ is a subset of the set of feasible interdiction plans. (We usually associate each constraint with a solution of the subproblem, but this representation is equivalent.) Given an interdiction plan $\hat{\mathbf{x}}$ suggested by the master problem, the subproblem should find a scalar $c(\hat{\mathbf{x}})$ and vector of penalties $\mathbf{v}(\hat{\mathbf{x}})$ such that conditions [3.3.1-2] hold. Assume that we have such a subproblem, denote it by [G-Sub($\hat{\mathbf{x}}$)]; Then, we have established **Algorithm 1** for [GSIP]. Note that X is discrete and bounded, and therefore finite, so convergence is guaranteed.

From **Corollary 3.2**, in order to validate conditions [3.3.1-2] we need to solve [G-Sub($\hat{\mathbf{x}}$)] exactly so that we know $f(\hat{\mathbf{x}})$ exactly. However, sometimes a sub-optimal solution of [G-Sub($\hat{\mathbf{x}}$)] is sufficient:

Proposition 3.4: *Assume that for any* $\hat{\mathbf{x}} \in X$ *such that* $f(\hat{\mathbf{x}}) > z^*$, *the scalar* $c(\hat{\mathbf{x}})$ *and the vector of penalties* $\mathbf{v}(\hat{\mathbf{x}})$ *satisfy*

$$z^* \;\; < \;\; c(\hat{\mathbf{x}}) - \mathbf{v}(\hat{\mathbf{x}})^T \hat{\mathbf{x}} \qquad\qquad\qquad [3.3.3], \quad and$$

$$f(\mathbf{x}) \;\; \geq \;\; c(\hat{\mathbf{x}}) - \mathbf{v}(\hat{\mathbf{x}})^T \mathbf{x} \qquad \forall\, \mathbf{x} \in X \qquad\qquad [3.3.4].$$

(The multipliers need not satisfy conditions [3.3.1-2]*.) And, for any* $\hat{\mathbf{x}} \in X$ *such that* $f(\hat{\mathbf{x}}) = z^*$, $c(\hat{\mathbf{x}})$ *and* $\mathbf{v}(\hat{\mathbf{x}})$ *satisfy conditions* [3.3.1-2]. *Then,* $z^* = z^{**}$ *(where* z *is defined by [GSIP] and* z^{**} *is defined by* [GSIP-1]*.)* ∎

Proof: Condition [3.3.3] ensures that $z(\hat{\mathbf{x}}) > z^*$ for any $\hat{\mathbf{x}} \in X$ such that $f(\hat{\mathbf{x}}) > z^*$, and for any $\hat{\mathbf{x}} \in X$ such that $f(\hat{\mathbf{x}}) = z^*$, conditions [3.3.1] and [3.3.2] ensure that $z(\hat{\mathbf{x}}) = z^*$.

Thus, $z^{**} = \min_{\mathbf{x} \in X} z(\mathbf{x}) = z^*$.

We now describe a possible use for the last proposition. Assume that during the solution process of [G-Sub($\hat{\mathbf{x}}$)] we know that $\hat{\mathbf{x}}$ cannot be optimal, i.e., we find $\mathbf{y}$ such that $g(\hat{\mathbf{x}}, \mathbf{y}) > \bar{z}$. Then, by **Proposition 3.4**, verifying that conditions [3.3.3-4] hold with respect to $\hat{\mathbf{x}}$ is sufficient to guarantee convergence. See an implementation of this idea in **Chapter IV**, in Procedure **Cutoff**.

Now that we have established the basics of **Algorithm 1** for [GSIP], we can modify, extend and improve the techniques as we did for MXSP and MILSIP:

(a)   Add all the enhancements discussed with regard to **Algorithm 1**, including the integrality cuts,

(b)   Derive **Algorithm 2** and **2E** (when the different enhancements are practical), and,

(c)   Finally, derive **Algorithm 3** for [GSIP].

We would like to point out that the existence of scalars and penalty multipliers that satisfy conditions [3.3.1-2] does not require that the function $f(\mathbf{x})$ be convex, or anything else, when the set X includes only binary variables. This is so because we are interested in the value of $f(\mathbf{x})$ only at certain of the corner points of the n-dimensional hypercube, and over these points it is possible to support any kind of function with a linear cut. (For instance, if we know that $z^* \geq 0$ we can always set $c(\hat{\mathbf{x}}) = f(\hat{\mathbf{x}})$ and for every k, let $v_k(\hat{\mathbf{x}}) = f(\hat{\mathbf{x}})$. The corresponding cut we add to the master problem,

$$z \geq c(\hat{\mathbf{x}}) - \sum_j x_j v_j(\hat{\mathbf{x}})\hat{y}_j \,,$$

would be valid, but useless.) However, it is usually easier to find relatively good penalty multipliers when the function $f(\mathbf{x})$ is convex for continuous $\mathbf{x}$, using gradient or sub-gradient information.

In general, **Proposition 3.3** and **Proposition 3.4** do not provide a method to derive the necessary penalty multipliers for Benders cuts (although the constant $c(\hat{\mathbf{x}})$ can always be found by solving [G-Sub $(\hat{\mathbf{x}})$]). However,, we exploit those propositions in the next two chapters to validate penalty multipliers that we can create based on the special structure of certain system-defense problems.

C. CONCLUSIONS

In this chapter we have shown that the techniques used to solve the shortest-path network-interdiction problem can be used for solving interdiction problems concerned with more complicated systems. Sufficient conditions are given to establish a Benders-type decomposition algorithm for solving a general system-interdiction problem, too.

The construction of a useful decomposition algorithm for solving a system-interdiction problem depends on the specific structure of the interdicted system. As shown in this chapter, the helpful enhancements for solving MXSP we saw in **Chapter II** may be applicable to other interdiction problems, as well.

In the next two chapters we use this chapter's results to construct algorithms for a system-defense problem and a stochastic system-interdiction problem. Both cases are based on MXSP, but just as we have shown in this chapter, the results there can be applied to more general system-defense and stochastic system-interdiction problems.

IV. SYSTEM DEFENSE – THE SHORTEST-PATH NETWORK DEFENSE PROBLEM

When a system user expects his system to be interdicted, he may be able to expend resources to protect that system to mitigate against the effects of interdiction. In this section we are interested in the following question: How should a system user employ limited resources to "harden" the components of his system to best protect against interdiction, given that the interdictor will optimize his interdiction with knowledge of those improvements? To answer to this question, we formulate and discuss the general system-defense problem (SD) and suggest extensions of **Algorithm 1-3** as solution procedures.

In the United States, the importance of system defense has been underscored by establishment of the President's Commission on Critical Infrastructure Protection (PCCIP). The executive order that creates the PCCIP (The White House 1996) states:

> "Certain national infrastructures are so vital that their incapacity
> or destruction would have a debilitating impact on the defense or
> economic security of the United States. These critical infrastructures
> include telecommunications, electrical power systems, gas and oil storage
> and transportation, banking and finance, transportation, water supply
> systems, ..."

The goal of PCCIP is to develop a strategy for protecting those systems against both physical and electronic attacks. The models we propose are most suitable for studying survivability of systems subject to physical attack.

Throughout the development of SD, we use the problem of *Defending the Shortest Path* (DSP) as an illustrative example. However, the results can be easily generalized to other and more general system defense problems.

A.   DEFENDING THE SHORTEST PATH – THE MODEL

DSP is an extension to MXSP (see **Chapter II**), where before the leader attempts any interdictions, the network user may harden (protect) some of his arcs against a possible interdiction. The network user has a fixed budget for hardening arcs and any hardened arc is considered invulnerable to subsequent interdiction. So, in DSP, the network user first hardens certain arcs, the leader then interdicts some subset of "unhardened" arcs, and finally the network user traverses a post-interdiction shortest path.

We assume that the network user has limited resources and that he cannot make his system completely invulnerable. Thus, the network user cannot completely protect any shortest *s-t* path. Let the set of feasible defense plans for the network user be given by $G = \{\mathbf{g} \in \{0,1\}^{|A|} \mid H\mathbf{g} \le \mathbf{h}\}$. We assume absolute protection so that $g_k = 1$ implies that arc k cannot be interdicted. DSP finds the optimal defense strategy for the network user, $\mathbf{g}^*$, and the value of the system, i.e., the length of the shortest path the network user is assured to have available for use.

Notice that for every feasible defense plan $\mathbf{g}$, the associated value of the system is given by a solution to an MXSP where the protected activities are invulnerable. And, recall that MXSP is NP-hard. Thus, we need to solve an NP-hard problem just to evaluate the objective function of a feasible solution to DSP. This fact leads to the following complexity result:

Proposition 4.1: *DSP is NP-hard and not known to be in NP.* ∎

Formally, given a graph $\mathcal{G}=(\mathcal{N},\mathcal{A})$, DSP is defined as the following problem for

the system user:

$$[\text{DSP}] \quad z_D = \min_{\mathbf{g}\in G} \ \max_{\mathbf{x}\in X(\mathbf{g})} \ \min_{\mathbf{y}\in Y(\mathbf{x})} \ \mathbf{c}^T\mathbf{y}$$

$$\text{where} \qquad G \ = \{\mathbf{g}\in\{0,1\}^{|\mathcal{A}|} \mid H\mathbf{g}\le\mathbf{h}\},$$

$$X(\mathbf{g}) \ = \left\{\mathbf{x}\in\{0,1\}^{|\mathcal{A}|} \mid R\mathbf{x}\le\mathbf{r}, 0\le\mathbf{x}\le 1-\mathbf{g}\right\}, \text{ and}$$

$$Y(\mathbf{x}) \ = \left\{\mathbf{y} \left| \begin{array}{l} \mathbf{y} \text{ is an incidence vector for an } s\text{-}t \\ \text{path that is feasible with respect to } \mathbf{x} \end{array} \right.\right\}.$$

Note that DSP is a min-max-min instance of the *Linear System-Defense Problem*

(LSDP), which we defined in **Chapter I** (since $Y(\mathbf{x})$ can be represented by a set of linear

flow-balance constraints and non-negativity restrictions). For modeling purposes we have

the following assumption:

Assumption 4.1: The sets G, $X(\mathbf{g})$ for all $\mathbf{g}\in G$ and $Y(\mathbf{x})$ for all $\mathbf{x}$

$\left\{\mathbf{x}\in\{0,1\}^{|\mathcal{A}|} \mid R\mathbf{x}\le\mathbf{r}, 0\le\mathbf{x}\le 1\right\}$ are not empty. ∎

DSP can be viewed as a min-max system-interdiction problem where the network

user is the leader and the interdictor is the follower. In DSP, the network user, now

called the "defender," minimizes the effectiveness of the interdictor's best possible

interdiction plan by choosing a defense plan that prevents, or "interdicts," some of the

interdictor's possible activities. This observation suggests solving DSP through a nested

decomposition algorithm. In particular, the master problem for DSP uses one of

Algorithms 1, 2 or **3**, except that the variables correspond to defense plans, rather than interdiction plans, and each subproblem solves an instance of MXSP by applying one of those algorithms.

B. NESTED DECOMPOSITION FOR SOLVING DSP

Let $z(\mathbf{g})$ be the length of the shortest-path the network user guarantees by defending with plan $\mathbf{g}$. Then, the network user's problem is equivalent to $\min\limits_{\mathbf{g} \in G} z(\mathbf{g})$.

By **Proposition 3.3**, to apply **Algorithm 1** to DSP, it suffices to have for any given defense plan $\hat{\mathbf{g}}$ a constant $c(\hat{\mathbf{g}})$ and a vector of penalties $\mathbf{v}_D$ such that:

(a) $z(\mathbf{g}) \geq c(\hat{\mathbf{g}}) - \mathbf{g}^T V_D \mathbf{x}(\hat{\mathbf{g}}) \quad \forall \, \mathbf{g} \in G,$ and

(b) $z(\hat{\mathbf{g}}) = c(\hat{\mathbf{g}}) - \hat{\mathbf{g}}^T V_D \mathbf{x}(\hat{\mathbf{g}}),$

where $V_D = \text{diag}(\mathbf{v}_D)$, and $\mathbf{x}(\hat{\mathbf{g}})$ is the optimal response of the interdictor to $\hat{\mathbf{g}}$.

Since $\hat{\mathbf{g}}^T V_D \mathbf{x}(\hat{\mathbf{g}}) = 0$ for all $\hat{\mathbf{g}} \in G$ (the interdictor cannot interdict a protected arc), based on **Corollary 3.2** we must set $c(\hat{\mathbf{g}}) = z(\hat{\mathbf{g}}) = \mathbf{c}^T \mathbf{y}(\mathbf{x}(\hat{\mathbf{g}}))$, where $\mathbf{y}(\mathbf{x}(\hat{\mathbf{g}}))$ denotes the shortest-path response of the network user given $\mathbf{x}(\hat{\mathbf{g}})$. Thus, we can calculate $c(\hat{\mathbf{g}})$ by solving the MXSP associated with defense plan $\hat{\mathbf{g}}$. (For simplicity, we assume that $\mathbf{y}(\mathbf{x}(\hat{\mathbf{g}}))$ and $\mathbf{x}(\hat{\mathbf{g}})$ are unique, but all results in this chapter can easily be generalized to allow multiple optimal responses.)

Assuming the existence of a valid penalty vector $\mathbf{v}_D$, we can solve DSP through *Nested Algorithm 1* (denoted by **NA-1**) where the master problem and subproblem are:

$[\text{D - Master}(\hat{G})] \qquad z_{\hat{G}} = \min\limits_{\mathbf{g} \in G} \quad z$

$$\text{s.t.} \quad z \geq \mathbf{c}^T \mathbf{y}(\mathbf{x}(\hat{\mathbf{g}})) - \mathbf{g}^T V_D \mathbf{x}(\hat{\mathbf{g}}) \quad \forall \, \hat{\mathbf{g}} \in \hat{G}$$

$$[\text{D - Sub}(\hat{g})] \quad z_{\hat{g}} = \max_{x \in X(\hat{g})} \quad \min_{y \in Y(x)} \quad c^T y$$

where

$$X(\hat{g}) = \left\{ x \in \{0,1\}^{|A|} \,\middle|\, Rx \le r, \, 0 \le x \le 1 - \hat{g} \right\}, \quad \text{and}$$

$$Y(x) = \left\{ y \,\middle|\, \begin{array}{l} y \text{ is an incidence vector for an } s\text{-}t \\ \text{path that is feasible with respect to } x \end{array} \right\},$$

where $\hat{G}$ is a subset of all the possible defense plans.

At every iteration of the decomposition algorithm, the master problem suggests a new defense plan g and update $\underline{z}_D$, and the subproblem solves the system-interdiction problem associated with g, adds the solution to $\hat{G}$, and updates $\bar{z}_D$, if appropriate. (The subproblem is simply an MXSP which is solved with **Algorithm 1**, or **2** or **3**.) If the solution of the master problem or the subproblem is ever repeated, we must have $\bar{z}_D = \underline{z}_D$ and the algorithm has converged. Therefore, the algorithm is theoretically guaranteed to converge, if the number of possible interdiction plans or defense plans is finite.

The remaining question is, of course, how to determine a valid penalty vector v_D. We will answer this question for DSP, as well for the more general LSDP, under the following assumption.

Assumption 4.2: *The feasible set of interdiction plans, X, is "closed" in the sense that any interdiction plan that is a part of a feasible interdiction plan is feasible too.*

This assumption is reasonable if interdictions consume non-negative quantities of resource and do not generate additional resource.

Proposition 4.2: *Let* $V_I = \text{diag}(\mathbf{v})$ *be a valid penalty matrix for a linear system interdiction problem LSIP:*

$$[\text{LSIP}] \qquad z_I = \min_{\mathbf{x} \in X} \; \max_{\mathbf{y} \in Y(\mathbf{x})} \; \mathbf{c}^T \mathbf{y}$$

where
$$X = \left\{ \mathbf{x} \in \{0,1\}^n \;\middle|\; R\mathbf{x} \leq \mathbf{r} \right\}, \quad \text{and}$$

$$Y(\mathbf{x}) = \left\{ \mathbf{y} \;\middle|\; A\mathbf{y} \leq \mathbf{b}, \; 0 \leq \mathbf{y} \leq U(\mathbf{1} - \mathbf{x}) \right\}.$$

i.e., (by **Proposition 3.2***)* [LSIP] *and* [LSIP-1] *have the same set of optimal solutions and the same objective function value, where:*

$$[\text{LSIP-1}] \qquad z_I = \min_{\mathbf{x} \in X} \; \max_{\mathbf{y} \in Y} \; \mathbf{c}^T \mathbf{y} - \mathbf{x}^T V_I \mathbf{y}$$

where
$$X = \left\{ \mathbf{x} \in \{0,1\}^n \;\middle|\; R\mathbf{x} \leq \mathbf{r} \right\}, \quad \text{and}$$

$$Y = \left\{ \mathbf{y} \;\middle|\; A\mathbf{y} \leq \mathbf{b}, \; 0 \leq \mathbf{y} \leq U \right\}.$$

Also, let [LSDP] *be the system-defense problem associated with* [LSIP]:

$$[\text{LSDP}] \quad z_D = \max_{\mathbf{g} \in G} \; \min_{\mathbf{x} \in X(\mathbf{g})} \; \max_{\mathbf{y} \in Y(\mathbf{x})} \; \mathbf{c}^T \mathbf{y}$$

where
$$S = \left\{ \mathbf{g} \in \{0,1\}^n \;\middle|\; H\mathbf{g} \leq \mathbf{h} \right\},$$

$$X(\mathbf{g}) = \left\{ \mathbf{x} \in \{0,1\}^n \;\middle|\; R\mathbf{x} \leq \mathbf{r}, 0 \leq \mathbf{x} \leq \mathbf{1} - \mathbf{g} \right\}, \quad \text{and}$$

$$Y(\mathbf{x}) = \left\{ \mathbf{y} \;\middle|\; A\mathbf{y} \leq \mathbf{b}, \; 0 \leq \mathbf{y} \leq U(\mathbf{1} - \mathbf{x}) \right\}.$$

Then, when **Assumption 4.2** *holds,* $V_D = V_I$ *is a valid penalty matrix for solving SD with algorithm NA-1.*

Proof: By **Proposition 3.3** and **Corollary 3.2**, it is sufficient to show that for all $\mathbf{x} \in X$ and $\hat{\mathbf{g}} \in G$, $z_D(\hat{\mathbf{g}}) = \mathbf{c}^T \hat{\mathbf{y}} \leq \mathbf{c}^T \mathbf{y}(\mathbf{x}) + \hat{\mathbf{g}}^T V_I \mathbf{x}$ where $\hat{\mathbf{y}} = \mathbf{y}(\mathbf{x}(\hat{\mathbf{g}}))$. Let $\mathbf{x}_{\bar{\mathbf{g}}}$ be the interdiction plan that interdicts an activity only if it is interdicted by interdiction plan $\mathbf{x}$

and not protected by $\mathbf{g}$. **Assumption 4.1** and the construction ensure that $\mathbf{x}_{\bar{\mathbf{g}}} \in X(\mathbf{g})$.

Clearly, $\mathbf{c}^T \mathbf{y}(\mathbf{x}_{\bar{\mathbf{g}}}) \geq z_D(\mathbf{g})$. Now, $\mathbf{y}(\mathbf{x}_{\bar{\mathbf{g}}}) \in Y(\mathbf{x})$ and due the optimality of $\mathbf{y}(\mathbf{x})$ in the inner

maximization in [LSIP-1] that is associated with $\mathbf{x}$, we must have

$\mathbf{c}^T \mathbf{y}(\mathbf{x}) \geq \mathbf{c}^T \mathbf{y}(\mathbf{x}_{\bar{\mathbf{g}}}) - \mathbf{x}^T V_I \mathbf{y}(\mathbf{x}_{\bar{\mathbf{g}}})$. Notice that $(\mathbf{x}^T V_I \mathbf{y}(\mathbf{x}_{\mathbf{g}}))_k > 0$ can happen only when

$\mathbf{y}(\mathbf{x}_{\bar{\mathbf{g}}})_k > 0$ and $x_k = 1$, but that implies $g_k = x_k = 1$. So, $\mathbf{x}^T V_I \mathbf{y}(\mathbf{x}_{\bar{\mathbf{g}}}) = \mathbf{g}^T V_I \mathbf{y}(\mathbf{x}_{\bar{\mathbf{g}}})$, and

$\mathbf{c}^T \mathbf{y}(\mathbf{x}) + \mathbf{g}^T V_I \mathbf{y}(\mathbf{x}_{\bar{\mathbf{g}}}) \geq \mathbf{c}^T \mathbf{y}(\mathbf{x}_{\bar{\mathbf{g}}})$. This finishes the proof because $\mathbf{c}^T \mathbf{y}(\mathbf{x}_{\bar{\mathbf{g}}}) \geq z_D(\mathbf{g})$, and

so $\mathbf{c}^T \mathbf{y}(\mathbf{x}) + \mathbf{g}^T V_I \mathbf{y}(\mathbf{x}_{\bar{\mathbf{g}}}) \geq z_D(\mathbf{g})$, too. ∎

Corollary 4.1: **Proposition 4.2** holds even when some or all of the variables $\mathbf{y}$
(the system's fundamental variables) are restricted to integer values, since nowhere in the
proof of that proposition is $\mathbf{y}$ is required to be continuous.

To better understand **Proposition 4.2**, consider its meaning in DSP. In this case,
interdicting an arc k increases the length of the arc by d_k, and hence cannot increase the
length of the shortest path by more than d_k. In the same sense, protecting this arc cannot
decrease the shortest path by more than the difference between the original length of the
arc and its length after interdiction, namely d_k. Indeed, **Proposition 4.2** proves that
$V_D = D$, where D=diag($\mathbf{d}$), will work.

We now focus on $[\mathrm{D\text{-}Sub}(\hat{\mathbf{g}})]$ in **NA-1** for DSP. We might need to solve this
problem many times and so it may be useful to have some kind of a "warm start," that
uses information from previous iterations. When solving $[\mathrm{D\text{-}Sub}(\hat{\mathbf{g}})]$, the master
problem is:

$$[\text{Sub}(\hat{\mathbf{g}}) \text{-} \text{Master}(\hat{Y})] \qquad z_{\hat{Y}} = \max_{\mathbf{x} \in X(\mathbf{g})} z$$

$$\text{s.t.} \quad z \le \mathbf{c}^T \hat{\mathbf{y}} + \mathbf{x}^T D \hat{\mathbf{y}} \quad \forall \, \hat{\mathbf{y}} \in \hat{Y}$$

Notice that all the cuts of the form $z \le \mathbf{c}^T \hat{\mathbf{y}} + \mathbf{x}^T D \hat{\mathbf{y}}$ are valid independent of the defense plan. Thus, as a warm start, we can begin a new iteration of the algorithm with the set $\hat{Y}$ from the end of the previous iteration.

In **NA-1** we also incorporate Procedure **Cutoff**. Recall that the smallest objective values from the subproblems (the interdictor's problem) solved so far is an upper bound on z_D, denoted by $\bar{z}_D$. In consecutive iterations, we can terminate solving a subproblem when, given $\hat{\mathbf{g}}$, the subproblem finds a sub-optimal solution $\mathbf{x}(\hat{\mathbf{g}}) \in X(\hat{\mathbf{g}})$ with optimal response $\mathbf{y}(\mathbf{x}(\hat{\mathbf{g}}))$ such that $\mathbf{c}^T \mathbf{y}(\mathbf{x}(\hat{\mathbf{g}})) > \bar{z}_D$. This technique uses **Proposition 3.4**: Once we recognize that the current defense plan $\hat{\mathbf{g}}$ is not optimal, the new cut generated from the subproblem need not be tight at $\hat{\mathbf{g}}$. Thus we stop solving the subproblem with a sub-optimal solution $\mathbf{x}(\hat{\mathbf{g}})$, add $\mathbf{x}(\hat{\mathbf{g}})$ to $\hat{X}$ and solve again the master defense problem. Convergence is guaranteed because the new Benders cut, $z \ge \mathbf{c}^T \mathbf{y}(\mathbf{x}(\hat{\mathbf{g}})) - \mathbf{g}^T W \mathbf{x}(\hat{\mathbf{g}})$ ensures that $\hat{\mathbf{g}}$ can no longer be the optimal solution of the master defense problem. (Recall that the solution of the master defense problem $z_{\hat{X}}$ is a lower bound. If $\hat{\mathbf{g}}$ were still optimal, we would have a contradiction since we cannot have $z_{\hat{X}} \ge \mathbf{c}^T \mathbf{y}(\mathbf{x}(\hat{\mathbf{g}})) - \mathbf{g}^T W \mathbf{x}(\hat{\mathbf{g}}) > \bar{z}_D$.)

Nested Algorithm 2E, denoted by **NA-2** and derived from **Algorithm 2E**, can be used to solve DSP (and other LSDPs) as well. In the highest level, the system user tries

to prevent interdiction plans suggested by the interdictor. For each new defense plan the interdictor solves the interdiction problem with **Algorithm 2E**. When the system user fails to prevent all the interdiction plans suggested so far by the interdictor, the algorithm terminates. The best defense plan the system user has tried until now is an optimal defense plan. The algorithm must converge if the number of possible interdiction plans or defense plans is finite.

Recall that **Algorithm 2E** includes several special procedures, namely **Local_Search**, **Compare** and **Lift**. The concept of local search in DSP translates into the generation of more than one interdiction plan (to be covered by a defense plan) per iteration. Fortunately, when we solve each subproblem we are actually suggesting interdiction plans and evaluating their objective function values (shortest-path lengths given the interdiction). We can keep this information and every interdiction plan with objective value higher than z_D (when the interdiction plan was exposed or after the lower bound was updated), can be introduced into the set of interdiction plans to be covered by any new defense plan.

Procedures **Compare** and **Lift**, as described in **Chapter II** with respect to MXSP, apply to DSP with essentially no change.

In **NA-2** we incorporate Procedure **Cutoff** just as we do in **NA-1**. This procedure is helpful when we have a good heuristic for the interdictor's problem, and we do have such a heuristic when the interdictor has a single resource constraint. In this case, we can solve the interdictor's master problem as an SCP ([Master($\hat{Y}$)-2c] in **Chapter II**), and we can run **Algorithm 2E** as a heuristic by solving that master problem using only the greedy SCP heuristic.

Last, as we suggested for **NA-1**, we can employ a warm start in **NA-2** using information from previous iterations. This may speed the convergence of the inner **Algorithm 2** used to solve the subproblem. Recall that at the end of **Algorithm 2E** we have two sets of *s-t* paths. Those sets are $\hat{Y}$ and Y^* where Y^* is a set of "reserve paths." We can merge these two sets and use them as a starting reserve set for the next iteration of the inner covering algorithm used to solve the subproblem. By doing this, we may require fewer iterations to generate a list of extreme points in $\hat{Y}$ that the interdictor cannot cover, and thus solve the subproblem using fewer iterations.

Unfortunately, computational experience indicates that the effort involved with the warm start is not always worthwhile. In MXSP we are able to generate paths very quickly, but we drop dominated paths and lift others, so the actual set $\hat{Y}$ at the end of the algorithm is only a small subset of the paths that were generated during the course of the algorithm. Therefore, if we want to save all the original paths, extra work is needed; it may be simpler and faster to regenerate those paths in subsequent iterations.

So far, we have discussed two nested decomposition algorithms for solving DSP, **NA-1** and **NA-2**. In should be clear that we can also establish an analogous version of **Algorithm 3**, *Nested Algorithm 3* **(NA-3)**. Given the above discussion creating this algorithm is straightforward and we omit any further description.

C. COMPUTATIONAL EXPERIENCE

To test the algorithms we have constructed, we use the same structure of the test problems of **Chapter II**. The only additional parameters are the total defense resource available h_0 (we assume a single defense resource) and the defense resource needed to defend each arc k, namely h_k, which is integer and uniformly distributed on $[1,h]$. Algorithm **NA-1** is tested with procedures **Warm_Start** and **Cutoff**. Algorithm **NA-2** is tested only with procedure **Cutoff**. We do not test **NA-3** here because for the amount of interdiction resources examined, **Algorithm 2E** and **Algorithm 3** give similar results (see **Table 2.1**).

Table 4.1 shows the average results across 10 problems for 3 different combinations of defense and interdiction resources. For instance, in **NA-2** problem set 3D, the master defense problem includes 535 cuts on average, each one representing an interdiction plan. The algorithm generates those 535 interdiction plans while solving only 265 interdiction problems, because of the **Local_Search** procedure. Moreover, those 155 interdiction problems are solved in 440 CPU seconds, even though a single interdiction problem with 30 units of interdiction resources requires 25 seconds CPU on average (see **Table 2.1** problem set 2.) The time improvement factor is almost 10, and is a result of procedure **Cutoff**.

Prob	h_0	r_0	Algorithm NA-1				Algorithm NA-2				
			T_1	S_1	N_1^D	N_1^I	T_2	S_2	N_2^D	N_2^I	P_2^D
1D	10	20	325	60	40	140	33	17	65	1210	155
2D	15	20	475	175	72	180	75	70	135	1670	305
3D	10	30	–	–	–	–	440	265	155	3375	535

Table 4.1: Computational results for the shortest-path network defense problem, with 10×10 inner nodes (a=396), c=10, d=10, r=5 and h=5.

Legend: T_i Running time in CPU seconds for Algorithm NA-i.

S_i Standard deviation in CPU seconds of T_i.

N_i^D Number of iterations in the master defense problem in NA-i.

N_i^I Total number of iterations in the master problem of the sub-problem (the interdiction problem) in NA-i.

P_2^D Number of interdiction plans (i.e., cuts) in the master defense problem in NA-2.

In **NA-1** we can see the advantage of procedure **Warm_Start**, too. In NA-1 problem set 2D, we solve 72 interdiction problems on average, with total of 180 master iterations in the interdiction problems. On the other hand, when we solve only one interdiction problem, we need 51 iterations on average (see **Table 3.1**). Thus, every new defense plan requires, on average, only 2 iterations in the interdiction problem, due to the **Warm_Start** and **Cutoff** procedures. Those 72 interdiction problems are solved in 475 CPU seconds while a single interdiction problem requires 110 CPU seconds (see **Table 3.1**). Therefore, the overall time improvement factor is more than 15.

D. CONCLUSIONS

The system-defense problem that we first defined as a natural extension for the system-interdiction problem, turns out to be a system-interdiction problem in itself, where the defender interdicts the interdictor's system. This observation let us solve the problem of defending the shortest path with nested decomposition algorithms. Fortunately:

(a)   We can use the fact that the defender interdicts a system-interdiction problem, to find valid penalty multipliers that are needed in **Algorithm 1**,

(b)   The enhancements included in **Algorithm 2E** are applicable here, too, and

(c)   Every subproblem solves a system-interdiction problem with a decomposition algorithm, but solving k subproblems doesn't require k times the time that one problem requires, due to **Warm_Start** and **Cutoff** procedures we include in the nested decomposition.

V. STOCHASTIC SHORTEST-PATH NETWORK INTERDICTION

Uncertainty may play a key role in some interdiction scenarios. For instance, the interdictor may have only limited intelligence on the system he attacks, and the success of interdiction attempts may be uncertain. Thus, the interdictor must determine his actions with incomplete information about the current state of the system and/or how the system will "react" after interdiction. We take an obvious approach to modeling stochastic situations: We assume that the interdictor has a measure for the expected value of the system after interdiction, and that he wishes to degrade this measure as much as possible.

Throughout the chapter we focus on the max-min stochastic shortest-path network-interdiction problem, S-MXSP, where interdiction success is uncertain. We note that Cormican *et al.* (1998) have studied one stochastic network-interdiction problem, with a different objective than MXSP, where other network data may be uncertain, too. However, we assume that all network data are known exactly. We show how the results of **Chapter III** can be used to establish decomposition algorithms for solving such stochastic network-interdiction problems, exactly or approximately. The approach can be used for solving other stochastic system-interdiction problems, too.

A.   THE MODEL

The mathematical programming formulation of S-MXSP on a directed graph $\mathcal{G}=(\mathcal{N},\mathcal{A})$ is:

$$[\text{S - MXSP}] \qquad \max_{\mathbf{x} \in X} E\left[\min_{\mathbf{y} \in Y} \sum_{k \in \mathcal{A}} (c_k + x_k \tilde{s}_k d_k) y_k \right]$$

where:

(a) $\tilde{\mathbf{s}} \in \{0,1\}^{|\mathcal{A}|}$ is a random vector: The outcome of the random variable $\tilde{s}_k$ is denoted by $\hat{s}_k$. $\hat{s}_k = 1$ with probability p_k, and $\hat{s}_k = 0$ with probability $1-p_k$. Thus, $0 \le p_k \le 1$ is the probability an interdiction attempt on arc k is successful. We assume that the successes of separate interdiction attempts are independent events.

(b) The rest of the formulation is the same as in MXSP. Thus, $X = \{\mathbf{x} \in \{0,1\}^{|\mathcal{A}|} \mid R\mathbf{x} \le \mathbf{r}\}$ represents the set of feasible interdiction plans and Y is the set of all s-t paths, represented in MXSP through flow balance constraints. For simplicity and without lost of generality we assume that for every k, $d_k < \infty$ (an infinite delay can be replaced by a very large, but finite, delay, which would ensure that no shortest path would use arc k when it is interdicted), and we assume that all arcs are interdictable (a non-interdictable arc k can be modeled by setting $d_k = 0$).

In S-MXSP, the inner minimization problem is a standard shortest-path problem with arc lengths $c_k + x_k \hat{s}_k d_k$. That is, the network user finds a shortest s-t path given the interdiction plan $\mathbf{x}$ and its random outcome $x_k \hat{s}_k$ for every k. We denote this shortest path by $\hat{\mathbf{y}}(\mathbf{x}, \hat{\mathbf{s}})$ where $\hat{y}_k(\mathbf{x}, \hat{\mathbf{s}}) = 1$ if the path $\hat{\mathbf{y}}(\mathbf{x}, \hat{\mathbf{s}})$ uses arc k, else $\hat{y}_k(\mathbf{x}, \hat{\mathbf{s}}) = 0$. For

simplicity, we assume $\hat{\mathbf{y}}(\mathbf{x}, \hat{\mathbf{s}})$ is unique, but all results in this chapter can easily be generalized to allow multiple shortest paths.

$$\text{Let} \quad E\left[\min_{\mathbf{y} \in Y} \sum_{k \in \mathbf{A}} (c_k + x_k \tilde{s}_k d_k) y_k\right] = E\left[\sum_{k \in \mathbf{A}} (c_k + x_k \tilde{s}_k d_k) \hat{y}_k(\mathbf{x}, \tilde{\mathbf{s}})\right] = z(\mathbf{x}) \quad \text{be the}$$

expected length of the shortest s-t path given interdiction plan $\mathbf{x}$, so that S-MXSP is equivalent to $\max_{\mathbf{x} \in X} z(\mathbf{x})$. $z(\mathbf{x})$ can be calculated exactly by solving the shortest-path problem associated with each outcome of the random vector $\tilde{\mathbf{s}}$. The literature also offers several algorithms for estimating $z(\mathbf{x})$ (e.g., Alexopoulos 1997, Fishman 1985).

The basic model assumes that only a single interdiction may be attempted on any arc. The following discussion shows that this is not actually a restriction. Assume that we have an arc k, which can be attacked by n different methods. Each method includes one or more independent and/or dependent interdiction attempts. For instance, arc k can be attacked by an airplane formation, 2 cruise missiles, or both, so we have three possible methods. Denote by p_k^j the probability of successful interdiction when method of attack j is chosen, and by d_k^j the delay expected on arc k when attack by method j is successful. To handle this situation we introduce the following construction:

(a)   We "break" the arc k into n serial arcs, $k^1, \ldots, k^n$, each with length c_k / n.

(b)   We set the probability of successful interdiction on arc k^j to be p_k^j and the delay when the arc is interdicted successfully to d_k^j.

(c)   Last, we add the constraint $x_{k_1^1} + x_{k_1^2} + \ldots + x_{k_1^n} \leq 1$ to make sure that only one method of attack is chosen (recall that a method of attack may include several interdiction attempts).

The construction ensures that if none of the arcs $k^1, \ldots, k^n$ is interdicted, or one of the arcs is interdicted but not successfully, the effective length of the composite arc k_1 is $n \times (c_k/n) = c_k$. But if method of attack j is chosen, and arc k^j is interdicted successfully, the effective length of the composite arc k_1 is $n \times (c_k/n) + d_k^j$, as required

B. DECOMPOSITION APPROACH

By **Corollary 3.2**, we can construct **Algorithm 1** for solving S-MXSP if we have constants $c(\hat{\mathbf{x}})$ and penalty multipliers $v_k(\hat{\mathbf{x}})$ such that:

[5.1] $\qquad\qquad z(\hat{\mathbf{x}}) \;=\; c(\hat{\mathbf{x}}) + \sum_{k \in \mathcal{A}} v_k(\hat{\mathbf{x}})\hat{x}_k \qquad \forall\, \hat{\mathbf{x}} \in X, \quad$ and

[5.2] $\qquad\qquad z(\mathbf{x}) \;\le\; c(\hat{\mathbf{x}}) + \sum_{k \in \mathcal{A}} v_k(\hat{\mathbf{x}})x_k \qquad \forall\, \hat{\mathbf{x}}, \mathbf{x} \in X$

Given the coefficients $c(\hat{\mathbf{x}})$ and $\mathbf{v}(\hat{\mathbf{x}})$, the master problem of **Algorithm 1** is

$\left[\text{Master}(\hat{X})\right] \qquad \max_{z,\mathbf{x}} \quad z$

$$\qquad\qquad \text{s.t.} \quad z \le c(\hat{\mathbf{x}}) + \sum_{k \in \mathcal{A}} v_k(\hat{\mathbf{x}})x_k \;\; \text{for every } \hat{\mathbf{x}} \in \hat{X}$$

where $\hat{X}$ is a subset of the set of feasible interdiction plans, and the subproblem, $\text{Sub}(\hat{\mathbf{x}})$, provides $c(\hat{\mathbf{x}})$ and $\mathbf{v}(\hat{\mathbf{x}})$ such that [5.1] and [5.2] hold. The following proposition shows how valid $c(\hat{\mathbf{x}})$ and $\mathbf{v}(\hat{\mathbf{x}})$ can be calculated.

Proposition 5.1: *For every* $\hat{\mathbf{x}} \in X$, *define*:

[5.3] $\qquad\qquad c(\hat{\mathbf{x}}) \;=\; \sum_{k \in \mathcal{A}} c_k E\left[\hat{y}_k(\hat{\mathbf{x}}, \tilde{s})\right], \qquad\qquad$ *and*

[5.4] $\qquad\qquad v_k(\hat{\mathbf{x}}) \;=\; d_k E\left[\tilde{s}_k \hat{y}_k(\hat{\mathbf{x}}, \tilde{s})\right] \qquad \forall k \in \mathcal{A}$

Then, conditions [5-1] *and* [5-2] *hold.*

Proof:

$$z(\mathbf{x}) \;=\; E\left[\min_{\mathbf{y}\in Y}\sum_{k\in\mathcal{A}}(c_k + x_k \tilde{s}_k d_k)y_k\right]$$

$$= E\left[\sum_{k\in\mathcal{A}}(c_k + x_k \tilde{s}_k d_k)\hat{y}_k(\mathbf{x},\tilde{\mathbf{s}})\right]$$

$$\le E\left[\sum_{k\in\mathcal{A}}(c_k + x_k \tilde{s}_k d_k)\hat{y}_k(\hat{\mathbf{x}},\tilde{\mathbf{s}})\right] \qquad \text{for all } \hat{\mathbf{x}}\in X$$

and equality must hold when $\hat{\mathbf{x}} = \mathbf{x}$, because given $\hat{\mathbf{x}}$ and $\hat{\mathbf{s}}$, $\hat{y}_k(\mathbf{x},\tilde{\mathbf{s}})$ is, by definition, the shortest path. Now,

$$E\left[\sum_{k\in\mathcal{A}}(c_k + x_k \tilde{s}_k d_k)\hat{y}_k(\hat{\mathbf{x}},\tilde{\mathbf{s}})\right]$$

$$= \sum_{k\in\mathcal{A}} E\left[c_k \hat{y}_k(\hat{\mathbf{x}},\tilde{\mathbf{s}})\right] + \sum_{k\in\mathcal{A}} x_k d_k E\left[\tilde{s}_k \hat{y}_k(\hat{\mathbf{x}},\tilde{\mathbf{s}})\right]$$

$$= c(\hat{\mathbf{x}}) + \sum_{k\in\mathcal{A}} v_k(\hat{\mathbf{x}})x_k, \qquad \blacksquare$$

Remarks:

(a) $c(\hat{\mathbf{x}})$ is the average length of the shortest path that the network user traverses, excluding delays, given interdiction plan $\hat{\mathbf{x}}$. $E\left[\hat{y}_k(\hat{\mathbf{x}},\tilde{\mathbf{s}})\right]$ is the probability that the network user traverses arc k, given interdiction plan $\hat{\mathbf{x}}$.

(b)   If $\hat{x}_k = 1$, $v_k(\hat{\mathbf{x}})$ is the average delay the network user experiences on arc k, given interdiction plan $\hat{\mathbf{x}}$. $E\left[\tilde{s}_k \hat{y}_k(\hat{\mathbf{x}},\tilde{\mathbf{s}})\right]$ is the probability that the network user traverses arc k given that the arc is successfully interdicted.

(c)   The proposition actually establishes that if $\hat{\mathbf{x}}$ does not interdict arc k, $v_k(\hat{\mathbf{x}})$ bounds the (average) gain the interdictor can achieve over $z(\hat{\mathbf{x}})$ by interdicting this arc. Notice that when $\hat{x}_k = 0$, $E\left[\tilde{s}_k \hat{y}_k(\hat{\mathbf{x}},\tilde{\mathbf{s}})\right] = E\left[\tilde{s}_k\right]E\left[\hat{y}_k(\hat{\mathbf{x}},\tilde{\mathbf{s}})\right]$

$= p_k E\big[\hat{y}_k(\hat{\mathbf{x}}, \tilde{\mathbf{s}})\big]$ where $E\big[\hat{y}_k(\hat{\mathbf{x}}, \tilde{\mathbf{s}})\big]$ is the probability that the network user traverses arc k given interdiction plan $\hat{\mathbf{x}}$.

Proposition 5.1 enables solution of S-MXSP through Benders decomposition, i.e., through **Algorithm 1**, at least in theory. The master problem suggests an interdiction plan $\hat{\mathbf{x}}$, and the subproblem (*i*) evaluates $z(\hat{\mathbf{x}})$, which may update the lower bound, and (*ii*) generates a new cut for the master problem through calculation of $E\big[\hat{y}_k(\hat{\mathbf{x}}, \tilde{\mathbf{s}})\big]$ and $E\big[\tilde{s}_k \hat{y}_k(\hat{\mathbf{x}}, \tilde{\mathbf{s}})\big]$. To compute these expected values, the subproblem solves $2^{\sum \hat{x}_k}$ shortest path problems, one for each possible outcome of the interdiction plan $\hat{\mathbf{x}}$.

In addition, we can use the Benders cut for building the integrality and covering cuts in the usual way (see **Chapter III**), establishing versions of **Algorithm 2E**, the covering decomposition, and **Algorithm 3**, the hybrid algorithm, for solving S-MXSP. An important part of those two algorithms is a local-search procedure that can find more than one covering cut per iteration. We will discuss this in more detail later.

A major difficulty of all three decomposition algorithms is the exponential complexity of the subproblem. For instance, if an interdiction plan interdicts 10 arcs, the subproblem requires solution of 1024 shortest-path problems, which our code does in less than 2 CPU seconds (for a 10×10 network). An average of 2 seconds CPU time for the subproblem is not a major concern. But, if the interdiction plan interdicts 20 arcs, the running time of the subproblem increases to over 2000 CPU seconds per iteration! Thus, the subproblem quickly becomes intractable.

.

C. APPROXIMATION THROUGH DECOMPOSITION

In trying to cope with the complexity of the subproblem, we suggest a series of approximation algorithms which share the following principles:

(a) There is no change in the master problem; either one of the three decomposition master problems may be used. But, to keep the discussion simple, we assume that the master problem of **Algorithm 1** is used.

(b) The subproblem solves a "reasonable number" of shortest-path problems.

(c) Given an interdiction plan $\hat{\mathbf{x}}$, the subproblem for approximation h, sets values for

$$c(\hat{\mathbf{x}}) = c^h(\hat{\mathbf{x}}) \quad \text{and} \quad v_k(\hat{\mathbf{x}}) = v_k^h(\hat{\mathbf{x}}) \quad \text{such} \quad \text{that} \quad [5\text{-}2] \quad \text{holds,} \quad \text{i.e.,}$$

$$z(\mathbf{x}) \le c^h(\hat{\mathbf{x}}) + \sum_{k \in \mathbf{A}} v_k^h(\hat{\mathbf{x}}) x_k \qquad \forall\, \hat{\mathbf{x}}, \mathbf{x} \in X. \qquad \text{As} \quad \text{a} \quad \text{direct} \quad \text{result,}$$

$$z^h(\hat{\mathbf{x}}) = c^h(\hat{\mathbf{x}}) + \sum_{k \in \mathcal{A}} v_k^h(\hat{\mathbf{x}}) \hat{x}_k \text{ is an upper bound for } z(\hat{\mathbf{x}}).$$

(d) We add to the master problem for approximation h the valid inequality, or "slack cut," $z \le c^h(\hat{\mathbf{x}}) + \sum_{k \in \mathcal{A}} v_k^h(\hat{\mathbf{x}}) x_k$. We call that a "slack cut" because [5-1] may not hold. However, since [5-2] holds, the value of the objective function of the master problem, denoted by $\bar{z}^h$ for approximation h, is a valid upper bound for the solution of S-MXSP. We may also add to the master problem the integrality cut associated with this slack cut.

(e) For a given master problem, we define $\underline{z}^h = \max_{\hat{\mathbf{x}} \in \hat{X}} z^h(\hat{\mathbf{x}})$. We use $\underline{z}^h$ as an artificial, possibly invalid, lower bound on $z(\mathbf{x}^*)$.

(f) When the difference between $\bar{z}^h$ and $\underline{z}^h$ drops below a designated *approximation gap*, we say that "the approximation algorithm has converged."

(g) The algorithm must converge because the number of feasible interdiction plans is finite, and in every iteration the master problem either suggests a new interdiction plan that is not yet in $\hat{X}$, or $\bar{z}^h = \underline{z}^h$.

(h) The procedure actually solves a problem with a modified objective $z^h(\hat{\mathbf{x}})$. Denote by $\mathbf{x}^h$ the optimal (or nearly so) solution that approximation algorithm h achieves using the modified objective.

(i) When practical, the algorithms establish a lower bound on the solution of S-MXSP and a *true optimality gap*, $\bar{z}^h - z(\mathbf{x}^h)$, by calculating $z(\mathbf{x}^h)$ using full enumeration. When this is impractical, the algorithms estimate $z(\mathbf{x}^h)$ through sampling. (Any other algorithm that gives a lower bound on $z(\mathbf{x}^h)$ could be used here, too.)

(j) Last, as an optional step, full enumeration of $z(\mathbf{x}^h)$ can be used to tighten the cut in the master problem associated with $\mathbf{x}^h$, and the master problem can be re-solved. This may improve the upper bound.

To establish an approximation algorithm as described above, we only require a subproblem that sets values for $c^h(\hat{\mathbf{x}})$ and $v_k^h(\hat{\mathbf{x}})$ such that [5-2] holds. Next, we introduce a proposition that helps us to derive such subproblems, but first we need some definitions.

Definition 5.1: A *deterministic reply strategy* $\mathbf{y}^h(\hat{\mathbf{s}})$ for the network user, is a function that assigns an *s-t* path $\mathbf{y}^h(\hat{\mathbf{s}})$, for all possible outcomes $\hat{\mathbf{s}} \in S$.

The deterministic reply strategy represents a possible way of action for the network user. Given outcome $\hat{\mathbf{s}}$, the reply strategy assumes that the network user traverses path $\mathbf{y}^h(\hat{\mathbf{s}})$, independently of the current interdiction plan. This is a feasible reply but the network user may have better choices. Therefore, for every interdiction plan $\mathbf{x}$, the average shortest-path length achieved by $\mathbf{y}^h(\hat{\mathbf{s}})$ given $\mathbf{x}$, is an upper bound on $z(\mathbf{x})$. We now define a more general set of feasible reply strategies.

Definition 5.2: A *randomized reply strategy* $\tilde{\mathbf{y}}^h(\hat{\mathbf{s}})$ for the network user is a function that assigns a probability distribution $\phi(\mathbf{y}|\hat{\mathbf{s}})$, over the set of all *s-t* paths (denoted by $\mathbf{y} \in Y$), for all possible outcomes $\hat{\mathbf{s}} \in S$. (Thus, $\sum_{\mathbf{y} \in Y} \phi(\mathbf{y}|\hat{\mathbf{s}}) = 1$ for all $\hat{\mathbf{s}} \in S$.)

$\tilde{\mathbf{y}}^h(\hat{\mathbf{s}})$ also represents a possible way of action for the network user. Given outcome $\hat{\mathbf{s}}$, the randomized reply strategy assumes that the network user traverses path $\hat{\mathbf{y}}$ with probability $\phi(\hat{\mathbf{y}}|\hat{\mathbf{s}})$, independently of the current interdiction plan. Note that any deterministic reply strategy can be defined as a randomized reply strategy, too. We now show formally how to devise a slack cut, and thus an approximation algorithm, from any reply strategy.

Proposition 5.2: Let $\tilde{\mathbf{y}}_{\hat{\mathbf{x}}}^{h}(\hat{\mathbf{s}})$ be a randomized reply strategy associated with approximation h. Define:

$$[5.5] \qquad c^{h}(\hat{\mathbf{x}}) = \sum_{k \in \mathcal{A}} c_{k} E\left[\sum_{\mathbf{y} \in Y} \phi_{\hat{\mathbf{x}}}(\mathbf{y}|\hat{\mathbf{s}}) y_{k}\right], \qquad \text{and}$$

$$[5.6] \qquad v_{k}^{h}(\hat{\mathbf{x}}) = d_{k} E\left[\tilde{s}_{k} \sum_{\mathbf{y} \in Y} \phi_{\hat{\mathbf{x}}}(\mathbf{y}|\hat{\mathbf{s}}) y_{k}\right] \qquad \forall k \in \mathcal{A}$$

Then, [5-2] holds, i.e., $z(\mathbf{x}) \le c^{h}(\hat{\mathbf{x}}) + \sum_{k \in \mathcal{A}} v_{k}^{h}(\hat{\mathbf{x}}) x_{k} \quad \forall \, \hat{\mathbf{x}}, \mathbf{x} \in X$.

Proof: For all $\mathbf{x} \in X$,

$$z(\mathbf{x}) = E\left[\min_{\mathbf{y} \in Y} \sum_{k \in \mathcal{A}} (c_{k} + x_{k} \tilde{s}_{k} d_{k}) y_{k}\right]$$

$$= E\left[\min_{\mathbf{y} \in Y} \sum_{k \in \mathcal{A}} \left((c_{k} + x_{k} \tilde{s}_{k} d_{k}) \sum_{\hat{\mathbf{y}} \in Y} \phi_{\hat{\mathbf{x}}}(\hat{\mathbf{y}}|\tilde{\mathbf{s}}) y_{k}\right)\right]$$

$$\le E\left[\sum_{k \in \mathcal{A}} \left((c_{k} + x_{k} \tilde{s}_{k} d_{k}) \sum_{\hat{\mathbf{y}} \in Y} \phi_{\hat{\mathbf{x}}}(\hat{\mathbf{y}}|\tilde{\mathbf{s}}) \hat{y}_{k}\right)\right]$$

$$= \sum_{k \in \mathcal{A}} c_{k} E\left[\sum_{\hat{\mathbf{y}} \in Y} \phi_{\hat{\mathbf{x}}}(\hat{\mathbf{y}}|\tilde{\mathbf{s}}) \hat{y}_{k}\right] + x_{k} d_{k} E\left[\tilde{s}_{k} \sum_{\hat{\mathbf{y}} \in Y} \phi_{\hat{\mathbf{x}}}(\hat{\mathbf{y}}|\tilde{\mathbf{s}}) \hat{y}_{k}\right]$$

$$= c^{h}(\hat{\mathbf{x}}) + \sum_{k \in \mathcal{A}} v_{k}^{h}(\hat{\mathbf{x}}) x_{k} \quad \blacksquare$$

Corollary 5.1: We can establish an approximation algorithm by defining how a subproblem sets a reply strategy, deterministic or randomized, for any given interdiction plan $\hat{\mathbf{x}}$. $\blacksquare$

Thus, in the approximation algorithms the subproblem takes in an interdiction plan $\hat{\mathbf{x}}$ and returns a randomized reply strategy $\tilde{\mathbf{y}}_{\hat{\mathbf{x}}}^h(\hat{s})$ or a deterministic reply strategy $\mathbf{y}_{\hat{\mathbf{x}}}^h(\hat{s})$. In the exact decomposition algorithm the subproblem does the same, but there it finds the best reply strategy (which is always deterministic), while in the approximation algorithm the subproblem finds a sub-optimal strategy. However, in order for the approximation algorithm to work effectively, it should find a relatively good reply strategy without too much computational effort. We next describe several approximation algorithms, each one characterized by its reply strategy.

Algorithm H1

The first approximation through decomposition uses a simple expected-value approach. Given interdiction plan $\hat{\mathbf{x}}$, let $\mathbf{y}^1(\hat{\mathbf{x}})$ be the shortest s-t path in the network using expected arc lengths, i.e., using arc lengths $c_k + \hat{x}_k p_k d_k$. Now, for all interdiction plans $\hat{\mathbf{x}}$, define the deterministic reply strategy by $\mathbf{y}_{\hat{\mathbf{x}}}^h(\hat{s}) = \mathbf{y}^1(\hat{\mathbf{x}})$ for all $\hat{s} \in S$. By **Corollary 5.1** we have established an approximation algorithm, Algorithm H1.

The modified objective in H1 (see principle (h)) assumes that the follower knows the chosen interdiction plan and all success probabilities, but not the actual outcome of the executed interdiction plan. This "restricts the recourse" of the follower (see Morton and Wood 1999) so we obtain an upper bound on the solution for S-MXSP, as we already know, from **Proposition 5.2**.

In this approximation, every subproblem solves just one shortest-path problem. Actually, approximation algorithm H1 is a decomposition for the deterministic interdiction problem with d_k replaced by $d_k p_k = d_k E\tilde{s}_k$. Therefore, we can apply the **Algorithm 2E** or **Algorithm 3** with the **Local_Search** procedure established for the deterministic case. The running time of the approximation H1 (without the evaluation of the exact objective function of the candidate solution; see principle (i) above) should be roughly the same as the running time for the deterministic interdiction algorithms.

Algorithm H2

As a refinement to H1, the modified objective of approximation H2 assumes that the follower knows the original interdiction plan, all success probabilities and also (and this is the difference between H1 and H2) the number of successful interdictions. However, the follower does not know which specific arcs were interdicted successfully.

Given an interdiction plan $\hat{\mathbf{x}}$, let n be the number of interdiction attempts. For $m = 0,...,n$, $p^{(m)}(\hat{\mathbf{x}})$ is the probability that $\hat{\mathbf{x}}$ results in m successful interdictions. Let $d_k^m = E[\tilde{s}_k d_k \,|\, m, \hat{\mathbf{x}}]$. Hence, $c_k + \hat{x}_k d_k^m$ is the expected length of arc k given $\hat{\mathbf{x}}$ and m. Let $\mathbf{y}^{2,m}(\hat{\mathbf{x}})$ be the shortest s-t path in the network with these arc lengths. Now, for all interdiction plans $\hat{\mathbf{x}}$, define the deterministic reply strategy by $\mathbf{y}_{\hat{\mathbf{x}}}^h(\hat{\mathbf{s}}) = \mathbf{y}^{2,m(\hat{\mathbf{s}},\hat{\mathbf{x}})}(\hat{\mathbf{x}})$ for all $\hat{\mathbf{s}} \in S$ where $m(\hat{\mathbf{s}},\hat{\mathbf{x}})$ if the number of successful interdictions $\hat{\mathbf{x}}$ causes given outcome $\hat{\mathbf{s}}$. By **Corollary 5.1** we have established an approximation algorithm H2.

In this approximation, every subproblem solves just n shortest-path problems. (Notice that $y^{2,0}(\hat{\mathbf{x}})$ is the shortest s-t path with no interdiction, which is independent of $\hat{\mathbf{x}}$.) However, we need to calculate $p^{(m)}(\hat{\mathbf{x}})$ and d_k^m. When $p_k = p$ for every $k \in A$, $p^{(m)}(\hat{\mathbf{x}})$ can be calculated directly from the binomial distribution, and the fact that $d_k^m = \dfrac{m}{n} d_k$. When different arcs may have different probabilities p_k, these parameters can be calculated in $O(|A|^2)$ time using the generating function

$$f(t) = \prod_{k \in A} (p_k t + (1 - p_k)).$$

Approximation H2 partitions the probability space according to the total number of successful interdiction attempts. Other partitioning schemes may be used too. For instance, a partition can be based on the success of a single interdiction, or on the number of successes in a group of arcs, etc. Furthermore, an algorithm can start with a crude partition and refine it later, until it reaches a desired optimality gap. This procedure is called "sequential approximation" in the stochastic programming literature (e.g. Kall *et al* 1988) and it was used by Cormican *et al.* (1998) for solving a stochastic max-flow network-interdiction problem. Algorithm H3 uses yet another partitioning scheme.

Algorithm H3

In this algorithm, the subproblem solves the shortest-path problem for a relatively few, "most likely" outcomes, and bounds the shortest-path length at all other possible outcomes, using the simple expected-value approach. Assume for instance that $\hat{\mathbf{x}}$ interdicts 10 arcs and the probability of success is 0.8 for every arc. Instead of solving 1024 shortest-path problems (as the exact decomposition algorithm would do), an

approximation algorithm H3 of "enumeration depth" 2 solves only the cases where 8 $(8 = 10 - 2)$ or more of the arcs are successfully interdicted. There are only 56 such outcomes but they cover about 68% of the probability space. Then, the approximation bounds the shortest-path length in the remaining 968 outcomes by finding one shortest-path which is optimal with respect to the average delays across those outcomes.

To establish approximation H3 of enumeration depth 2 formally, define the following sets:

(a) S' is the set of all outcomes $\hat{s}$ with <u>2 or fewer</u> failures in $\hat{x}$, and

(b) S'' is the set of all outcomes $\hat{s}$ with <u>more than 2</u> failures in $\hat{x}$.

Now, for all interdiction plans $\hat{x}$, define the deterministic reply strategy by $y^h(\hat{s}) = y(\hat{x}, \hat{s})$ for all $s \in S'$, and by $y^h(\hat{s}) = y^3(\hat{x})$ for all $s \in S''$, where $y^3(\hat{x})$ is the shortest path with arcs lengths defined as $E\left[c_k + \hat{x}_k \tilde{s}_k d_k \middle| s \in S''\right]$ for all $k \in \mathcal{A}$.

Example 5.1

This example demonstrates the way subproblems work in the exact decomposition algorithm and in the three approximations suggested so far. Moreover, we later use this example to motivate a possible improvement to approximation H3.

Assume that the interdictor interdicts 4 arcs, denoted by $k1$, $k2$, $k3$, and $k4$. There are $2^4 = 16$ possible outcomes and each is denoted by a four-digit binary number. For instance, 1010 represents an outcome in which the interdiction of $k1$ and $k3$ are successful and the interdiction of $k2$ and $k4$ are not.

All the algorithms solve several shortest-path problems, each associated with a specific possible outcomes, or an average of several outcomes (combined by an

approximation algorithm). **Figure 5.1** shows how the different algorithms partition these 16 possibilities. In the figure, every cell represents group of outcomes that the algorithm links, and so every cell represents one shortest-path problem that the subproblem solves.

■

Exact Alg.	Approx. Alg. H1	Approx. Alg. H2	Approx. Alg. H3 (depth 2)
1111	1111	1111	1111
1110	1110	1110	1110
1101	1101	1101	1101
1011	1011	1011	1011
0111	0111	0111	0111
1100	1100	1100	1100
1010	1010	1010	1010
1001	1001	1001	1001
0110	0110	0110	0110
0101	0101	0101	0101
0011	0011	0011	0011
1000	1000	1000	1000
0100	0100	0100	0100
0010	0010	0010	0010
0001	0001	0001	0001
0000	0000	0000	0000

Figure 5.1: Given an interdiction plan with 4 attempts, every algorithm for S-MXSP partitions the 16 possible outcomes into a different set of "cells" (divided by horizontal lines), and solves one shortest-path problem for each cell.

Algorithm H4

In approximation H3, the reply strategy for each one of the outcomes in S'' is the same path, in particular, the path that is shortest on average over all outcomes in S''. (In the example of **Figure 5.1**, S'' is the large cell in the bottom of the list associated with H3.) The main idea of approximation H4 is to use information about shortest-path problems associated with outcomes in S' (in the example of **Figure 5.1**, those outcomes that are in separate cells in the top part of the list associated with H3) to define a possibly better reply strategy for outcomes in S''.

For instance, consider outcome 0100 in **Example 5.1**. Intuitively, it seems likely that shortest paths that are associated with outcomes 1100, 0110 and 0101 are relatively short paths with respect to outcome 0100. Given outcome 0100, none of these 3 paths is necessarily the shortest path, but they may be better, i.e., shorter on average, than the single path that is good on average with respect to all the outcomes in S''. In approximation H4, the reply strategy for each of the outcomes in S'' (the same set as in H3) is a randomized combination of optimal replies to related outcomes in S'.

To define precisely approximation H4 of enumeration depth 2, we first assume that $p_k = p$ for all arcs k. Let n be the number of interdiction attempts for a given interdiction plan $\hat{\mathbf{x}}$. For $m = 0,\ldots,n$, $p^{(m)}(\hat{\mathbf{x}})$ is the probability that $\hat{\mathbf{x}}$ results in m successful interdictions (given by the binomial distribution), and $S^m(\hat{\mathbf{x}})$ is the set of all outcomes with exactly m successful attempts in $\hat{\mathbf{x}}$. Now, given an interdiction plan $\hat{\mathbf{x}}$, we define the randomized reply strategy by defining $\phi_{\hat{\mathbf{x}}}(\mathbf{y}\,|\,\hat{\mathbf{s}})$ for all $\hat{\mathbf{s}} \in S$:

(a)   For all $\hat{\mathbf{s}} \in S^n(\hat{\mathbf{x}}) \cup S^{n-1}(\hat{\mathbf{x}}) \cup S^{n-2}(\hat{\mathbf{x}})$, let $\phi_{\hat{\mathbf{x}}}(\mathbf{y}|\hat{\mathbf{s}})$ be 1 for $\hat{\mathbf{y}}(\hat{\mathbf{x}},\hat{\mathbf{s}})$ and 0 otherwise.

(b)   For $m = 0,\ldots,n-3$ and all $\hat{\mathbf{s}} \in S^m(\hat{\mathbf{x}})$ we first identify the $\binom{n-m}{2}$ elements of $S^{n-2}(\hat{\mathbf{x}})$ where the same m attempts are successful too, denoted by $s^1, s^2, \ldots, s^{\binom{n-m}{2}}$. (Out of the $n-m$ interdiction failures in $\hat{\mathbf{x}}$ corresponding to $\hat{\mathbf{s}} \in S^m(\hat{\mathbf{x}})$, we need all combinations of two failures, each one corresponding to a different $\hat{\mathbf{s}} \in S^{n-2}(\hat{\mathbf{x}})$.) Let,

$$\phi_{\hat{\mathbf{x}}}(\mathbf{y}|\hat{\mathbf{s}}) = \begin{cases} 1/\binom{n-m}{2} & \mathbf{y} \in \hat{\mathbf{y}}\left\{\hat{\mathbf{x}}, \mathbf{s}^1), \hat{\mathbf{y}}(\hat{\mathbf{x}}, \mathbf{s}^2), \ldots, \hat{\mathbf{y}}(\hat{\mathbf{x}}, \mathbf{s}^{\binom{n-m}{2}})\right\} \\[2em] 0 & \text{otherwise} \end{cases}$$

By **Proposition 5.2** and **Corollary 5.1**, to establish an approximation algorithm the subproblem must calculate $E[\hat{y}_k(\hat{\mathbf{x}}, \tilde{s})]$ and $E[\tilde{s}_k \hat{y}_k(\hat{\mathbf{x}}, \tilde{s})]$ for all $k \in \mathcal{A}$. The following proposition shows how that can be done with respect to our definitions of approximation H4.

Proposition 5.3: *In approximation algorithm H4 of depth 2, for all $k \in \mathcal{A}$,*

[5.7] $E[\hat{y}_k(\hat{\mathbf{x}}, \tilde{s})] = \displaystyle\sum_{m=n-2}^{n} p^{(m)}(\hat{\mathbf{x}}) E\left[\hat{y}_k(\hat{\mathbf{x}}, \tilde{s}) \big| \mathbf{s} \in S^m(\hat{\mathbf{x}})\right] + \sum_{m=0}^{n-3} p^{(m)}(\hat{\mathbf{x}}) E\left[\hat{y}_k(\hat{\mathbf{x}}, \tilde{s}) \big| \mathbf{s} \in S^{n-2}(\hat{\mathbf{x}})\right]$

[5.8] $E[\tilde{s}_k \hat{y}_k(\hat{\mathbf{x}}, \tilde{s})] = \displaystyle\sum_{m=n-2}^{n} p^{(m)}(\hat{\mathbf{x}}) E\left[\tilde{s}_k \hat{y}_k(\hat{\mathbf{x}}, \tilde{s}) \big| \mathbf{s} \in S^m(\hat{\mathbf{x}})\right]$

$+ \displaystyle\sum_{m=0}^{n-3} p^{(m)}(\hat{\mathbf{x}}) E\left[\tilde{s}_k \big| \mathbf{s} \in S^m(\hat{\mathbf{x}})\right] E\left[\hat{y}_k(\hat{\mathbf{x}}, \tilde{s}) \big| \mathbf{s} \in S^{n-2}(\hat{\mathbf{x}}), \hat{s}_k = 1\right]$

Proof: By definition, for all $k \in \mathcal{A}$

$$E\left[\hat{y}_k(\hat{\mathbf{x}}, \tilde{s})\right] = \sum_{m=0}^{n} p^{(m)}(\hat{\mathbf{x}}) E\left[\hat{y}_k(\hat{\mathbf{x}}, \tilde{s}) \Big| s \in S^m(\hat{\mathbf{x}})\right] \quad \text{and}$$

$$E\left[\tilde{s}_k \hat{y}_k(\hat{\mathbf{x}}, \tilde{s})\right] = \sum_{m=0}^{n} p^{(m)}(\hat{\mathbf{x}}) E\left[\tilde{s}_k \hat{y}_k(\hat{\mathbf{x}}, \tilde{s}) \Big| s \in S^m(\hat{\mathbf{x}})\right],$$

and so it is enough to show that for all $m \le n - 3$ and for all $k \in \mathcal{A}$

$$E\left[\hat{y}_k(\hat{\mathbf{x}}, \tilde{s}) \Big| s \in S^m\right] = E\left[\hat{y}_k(\hat{\mathbf{x}}, \tilde{s}) \Big| s \in S^{n-2}\right] \quad \text{and}$$

$$E\left[\tilde{s}_k \hat{y}_k(\hat{\mathbf{x}}, \tilde{s}) \Big| s \in S^m\right] = E\left[\tilde{s}_k \Big| s \in S^m(\hat{\mathbf{x}})\right] E\left[\hat{y}_k(\hat{\mathbf{x}}, \tilde{s}) \Big| s \in S^{n-2}(\hat{\mathbf{x}}), \hat{s}_k = 1\right].$$

But, these are straightforward results of our definition of the reply strategy in approximation algorithm H4 of depth 2 and the assumption that $p_k = p$ for all k ∎

Remark: We conjecture that a modified reply strategy can be defined such that **Proposition 5.3** holds even if the probability that an interdiction attempt is successful is not the same for all interdictable arcs. That modified strategy needs to assign different weight to different paths in the definition of the reply strategy for $s \in S^m$, $m \le n - 3$, in order to keep [5.7] and [5.8] valid.

In order to calculate $E\left[\hat{y}_k(\hat{\mathbf{x}}, \tilde{s})\right]$ and $E\left[\tilde{s}_k \hat{y}_k(\hat{\mathbf{x}}, \tilde{s})\right]$ for all $k \in \mathcal{A}$ it is enough to calculate the probability of each outcome $\hat{s} \in S^n(\hat{\mathbf{x}}) \cup S^{n-1}(\hat{\mathbf{x}}) \cup S^{n-2}(\hat{\mathbf{x}})$, and to find the shortest-path given this $\hat{s}$, since for all $k \in \mathcal{A}$:

(a) $\quad \sum_{m=0}^{n-3} p^{(m)}(\hat{\mathbf{x}}) = 1 - \sum_{m=n-2}^{n} p^{(m)}(\hat{\mathbf{x}})$, and

(b) $\quad \sum_{m=0}^{n-3} p^{(m)}(\hat{\mathbf{x}}) E\left[\tilde{s}_k \Big| s \in S^m(\hat{\mathbf{x}})\right] = p_k - \sum_{m=m-2}^{n} p^{(m)}(\hat{\mathbf{x}}) E\left[\tilde{s}_k \Big| s \in S^m(\hat{\mathbf{x}})\right].$

D. A LOCAL-SEARCH PROCEDURE

Our experience with the different decomposition algorithms for solving the deterministic MXSP, as well as preliminary computational results for S-MXSP (reported later), indicate that the running time of the master problem is very sensitive to the number of Benders cuts. In fact, even though the subproblem of S-MXSP requires more work compared to the subproblem in MXSP, in both cases the limits of the decomposition algorithms are determined by the difficulty in solving the master problems, at least with our current technology.

In order to solve the problem with fewer master iterations (every iteration adds one more Benders cut), the hybrid algorithm can use a local-search procedure to generate more than one covering cut per iteration. Those cuts are added to the master problem, and tighten it. This discussion suggests that a local-search procedure may also be helpful for the different decomposition algorithms (exact or approximate) for S-MXSP. We describe a possible local-search procedure for the exact decomposition algorithm. However, the same approach may be use to develop a local-search procedure for all the approximation algorithms, too.

Let $\bar{y}_{\hat{x}}(\hat{s})$ be any reply strategy, deterministic or randomized, that the subproblem can define given interdiction plan $\hat{x}$. It may be, for instance, that $\bar{y}_{\hat{x}}(\hat{s})$ is the second shortest path given $\hat{x}$ and $\hat{s}$, or the second shortest path in the list of paths generated by our **Local_Search** procedure we described for MXSP. Then, by Proposition 5.2 we can define $\bar{c}(\hat{x}) = \sum_{k \in \mathcal{A}} E\left[c_k \bar{y}(\hat{x}, \tilde{s}_k)\right]$ and $\bar{v}_k(\hat{x}) = d_k E\left[\tilde{s}_k \bar{y}_k(\hat{x}, \tilde{s})\right]$, to obtain the valid Benders cut $z \leq \bar{c}(\hat{x}) + \sum_{k \in \mathcal{A}} \bar{v}_k(\hat{x}) x_k$ But, as we discussed earlier, too many Benders cut is

not a good idea. Therefore, the local-search procedure might only generate a covering cut based on this (slack) Benders cut, and add it to the master problem. (If $\bar{c}(\hat{\mathbf{x}})$ exceeds the current value of $\underline{z}$, we will not include this cut in the master problem, but rather, as in **Algorithm 2E** for MXSP, put it aside for possible later use.)

For instance, suppose we choose $\bar{y}_k(\hat{\mathbf{x}},\hat{\mathbf{s}})$ as the second shortest path given $\hat{\mathbf{x}}$ and $\hat{\mathbf{s}}$. Then, when we compute the exact Benders cut we also find the second shortest path in each of the subproblems. Those second shortest paths are used to compute a slack cut, and the covering cut associated with this slack cut is added to the master problem. Note that we can generate more than one slack cut per iteration by finding additional paths for every subproblem.

E. COMPUTATIONAL EXPERIENCE

To test the different algorithms, we use the shortest-path network and computational platform described in **Chapter II**. However, a new vector **p** is added, where p_k is the probability an interdiction attempt on arc k is successful. For the results reported here, we use the same success probability for every arc, denoted by p.

The following tables summarize the results for several different cases. We note that these are preliminary results and that none of the algorithms includes a local-search procedure. Thus, the exact and approximation decompositions are accomplished with **Algorithm 1**. In the case of approximation H1, we could have used the **Local_Search** procedure developed for deterministic network interdiction (that might have reduced its running time significantly, as seen in **Chapter II**) but we did not, in order to allow a fair comparison between this algorithm and the others.

Table 5.1 compares approximation H1, approximation H2 and the exact algorithm. In both approximations we use a 1% approximation gap, but, as the table shows, that translates to a 12-20% true optimality gap for our test problems. (See principles (f) and (i) in Section 5.C) for the definitions of approximation gap and optimality gap.)

We solve the same problems with the exact algorithm up to a 12% or 15% optimality gap. **Table 5.1** shows that the exact algorithm has similar running times to the approximations for this parameter setting. But, since computational effort in the exact algorithm is clearly more sensitive to the total amount of interdiction resource (we do not see this in the table because r_0 is relatively small), the approximation algorithm would probably outperform the exact algorithm with larger values of r_0. In any case, the running times indicate that we cannot successfully increase r_0 significantly, even in the approximations, without adding a local-search procedure or other effective enhancement.

H2 takes much more running time than H1, but yields a similar optimality gap. The major disadvantage of H2 is in the master problem level (the extra work in the subproblem is insignificant here.) The cuts in the master problem of H2 are less effective (compare the number of iterations the two algorithms need for convergence, N_{H2} versus N_{H1}), probably because every cut in H2 represents a weighted combination of multiple s-t paths, and not a single path. Thus, in the cuts of H2 the non-zero coefficients are smaller on average, and so the cuts are "flatter," (in the geometric sense) compared to the cuts in H1. (Notice that each cut in the exact algorithm is a weighted combination of many s-t paths and so is even flatter. As a result, the exact algorithm requires more iterations to

achieve a 12% gap, than approximations H1 and H2 need to achieve a 1% approximation gap, which is actually 1% optimality gap with respect to the modified objective.)

Note also that:

(a) The optimality gap of both approximations H1 and H2 becomes smaller, though not small enough, when p increases, and

(b) The optimality gap of all approximations increases when we have more interdiction resource.

(c) For the case $p = 0.7$ and $r_0 = 15$ (problem set 1S), we used the exact algorithm to solve the same problem to a 2% optimality gap. It turns out that the optimality gap obtained by the approximation algorithms (and the exact algorithm when used with a 15% gap) depends primarily on poor upper bounds rather than poor interdiction plans x. About 20% of the gap is due to the difference between the optimal objective value and the lower bound (i.e., the objective value of the incumbent solution), while the weak upper bound is responsible for 80% of the gap.

Problem	p	r_0	Approximation H1					Approximation H2					Exact Algorithm				
			LB_{H1}	UB_{H1}	$\%g_{H1}$	T_{H1}	N_{H1}	LB_{H2}	UB_{H2}	$\%g_{H2}$	T_{H2}	N_{H2}	LB_E	UB_E	$\%g_E$	T_E	N_E
1S	0.7	15	17.8	21.0	18.3	15	20	17.8	20.8	16.5	38	33	18.0	20.6	15	14	22
2S	0.7	20	18.4	22.0	20.1	84	33	18.4	21.8	18.6	243	54	18.5	21.3	15	87	37
3S	0.8	15	18.8	21.2	12.6	21	24	18.8	21.1	12.3	36	21	18.9	21.0	12	39	32
4S	0.8	20	19.5	22.5	15.6	102	39	19.5	22.4	14.4	266	53	19.5	21.7	12	255	60

Table 5.1: Computational results for S-MXSP. The test network has $64=8\times8$ inner nodes (a=238), c=10, d=10 and r=5. The approximation gap for H1 and H2 is 1%.

Legend: LB_h Lower bound achieved by Algorithm h.
           UB_h Upper bound achieved by Algorithm h.
           $\%g_h$ True optimality gap achieved by the Algorithm h.
           T_h Running time in CPU seconds for Algorithm h.
           N_h Number of iterations for Algorithm h.

Table 5.2 compares approximation H3, approximation H4 and the exact algorithm (both approximations are of enumeration depth 2). We solve the exact algorithm with 5% and 10% optimality gaps, and in order to obtain similar results we set the approximation gap to 1% and 4% in H3, and to 4% and 8% in H4.

The table shows that approximation H3 cannot establish small optimality gaps. In problems 7S we solve approximation H3 with a 1% approximation gap but obtain an 8% true optimality gap. On the same problems, approximation H4 with a 4% approximation gap yields a true optimality gap of 6.5%. Thus, approximation H4 estimates the reply to outcomes with 3 or more failures better than approximation H3, and so the modified objective of H4 is closer in its value to the optimal objective.

Table **5.2** also demonstrates, just as **Table 5.1**, that the approximations and exact algorithm have similar running times, when the exact algorithm is solved with a similar true optimality gap. But, with increased r_0 the approximations would probably outperform the exact algorithm.

Prob.	p	r_0	Approximation H3						Approximation H4						Exact Algorithm				
			%g	LB_{H3}	UB_{H3}	$\%g_{H3}$	T_{H3}	N_{H3}	%g	LB_{H4}	UB_{H4}	$\%g_{H4}$	T_{H4}	N_{H4}	LB_E	UB_E	$\%g_E$	T_E	N_E
5S	0.75	15	4.0	18.4	20.5	11.1	87	57	8.0	18.5	20.4	10.2	61	50	18.4	20.2	10	74	46
6S	0.75	20	4.0	19.2	21.9	14.5	200	73	8.0	19.1	21.3	11.8	567	97	19.1	21.0	10	550	87
7S	0.75	15	1.0	18.5	20.0	8.0	333	197	4.0	18.5	19.7	6.5	352	117	18.6	19.5	5	344	107

Table 5.2: Computational results for S-MXSP. The test network has 64=8×8 inner nodes (a=238), c=10, d=10 and r=5.

        Legend: %g   Approximation gap.
                   All other legend data as in **Table 5.1**

F.   CONCLUSIONS

In this chapter we have shown how our approach for solving deterministic system-interdiction problems can be extended to solve one type of stochastic system-interdiction problem. We revisited the shortest-path network-interdiction problem to demonstrate this, but this time assumed that interdiction success is uncertain. As expected, the stochastic problem is much more difficult to solve than the deterministic one, mainly because the subproblems in our decomposition algorithm generate "flat" cuts, and so the master problem requires more iterations to converge. Surprisingly, this is even true for an exact algorithm with exponential complexity in the subproblems. From

our experience with the deterministic interdiction problem (and the shortest-path network-defense problem) we know that a local-search procedure will likely accelerate the decomposition algorithms for the stochastic problems, but we have not yet implemented such a procedure.

We compared the exact decomposition algorithm to several approximation (decomposition) algorithms. On our test problems, the exact and approximation algorithms require similar running times (when the problem is solved to the same optimality gap) but in larger problems the approximations are likely to be better, because they don't require an exponential amount of work in the subproblem phase.

Among the approximations, algorithm H4 gives the best results. For a given optimality gap, running times for approximation H4 are similar to those of the other approximations, and of all the approximations tested, approximation H4 establishes the smallest optimality gaps. Approximation H4 finds the optimal reply of the network user for the most likely outcomes out of the interdiction attempts, and use that to approximate (and bound) the optimal network user replies for all other outcomes.

Last, we note that our approach for solving S-MXSP can be easily applied to other interdiction problems, where the success of each interdiction attempt is uncertain.

VI. CONCLUSIONS

This chapter reviews the accomplishments of this dissertation and suggests opportunities for further research.

In this dissertation we have discussed several problems concerning system interdiction and defense, using a shortest-path network-interdiction scenario to demonstrate our approach. We have addressed the following questions:

(a)   What is the best interdiction plan?

(b)   What is the best defense plan against a prospective set of interdictions?

(c)   What is the best interdiction plan when interdiction attempts might fail?

The deterministic shortest-path network-interdiction problem (MXSP) is discussed in **Chapter II**. MXSP assumes that a network user traverses a shortest path given the results of a prior interdiction, and the question is "What interdiction plan will maximize the length of that shortest path?" When interdiction of an arc increases its effective length by a finite amount (called "delay"), and the network user traverses the shortest path given the interdiction, we have shown how to formulate the problem as a mixed-integer program (MIP), and how to solve the problem with Benders decomposition. However, when interdiction of an arc makes the arc impassable, those solution techniques can be ineffective, and we therefore devised a second decomposition algorithm, in which the Benders master problem is replaced by a set-covering problem. Last, we combined the first two decomposition algorithms into a hybrid decomposition algorithm which gives the best computational results. All tests were performed on randomly generated networks, it would be interesting to repeat those tests on more realistic problems.

The hybrid algorithm includes several special enhancements, derived through the first two decomposition algorithms, especially (*i*) integrality cuts for the Benders master problem along with a method to tighten those cuts, and, (*ii*) covering constraints—which are best viewed as integrality cuts in this context—and a method to generate and lift them. Those enhancements were shown to be effective for solving MXSP. It would be interesting, when delays are finite, to see if the integrality cuts and/or covering constraints derived from the decompositions would be useful as (integrality) cuts for reducing solution times for MXSP solved as a MIP. It might also be possible to add some constraints, possibly aggregated, from the MIP to the decomposition master problems to tighten their relaxations and thereby improve solution times.

In **Chapter III** we showed how the techniques used to solve the shortest-path network-interdiction problem can be used for solving other interdiction problems where an interdictor tries to reduce the effectiveness of an adversary's system through interdiction. Thus, our methods can be used to interdict a shortest-path system with side constraints, disrupt activities in a PERT network in order to maximize project completion time, reduce the effectiveness of an economic system modeled as an optimization problem, etc.

In a wider perspective, the special enhancements we suggest for the basic master problem in Benders decomposition, i.e., the integrality and covering cuts, may be helpful while applying Benders decomposition to other problems with binary "complicating" variables. Consider, for example, a problem of the design and operation of a production and distribution system. These problems often involve (*i*) "strategic variables," which constitute binary decisions over facility locations and other issues of infrastructure, and,

(*ii*) "operational variables," usually assumed continuous. A common way of solving those problems is through Benders decomposition (e.g., Brown *et al.* 1987), where the subproblem is a (relatively simple) operational problem and the master problem deals with the binary strategic variables. It might be possible to improve running times of such decompositions by adding the integrality cuts and/or covering constraints to the Benders master problem, as was done with MXSP.

In **Chapter IV** we discussed a system-defense problem in which the system user can defend some of his activities, resources, etc., against prospective interdiction. The system-defense problem turns out to be a system-interdiction problem in itself, where the defender interdicts the interdictor's system. This view leads us to solve the problem of defending the shortest path (in this problem the network user defends some of the arcs in a shortest-path network and afterward the interdictor finds the best interdiction plan on the undefended arcs of the network) with nested decomposition algorithms. The algorithms we use include the enhancements developed for MXSP (and adapted for more general system-interdiction problems) and special procedures that take advantage of the structure of the nested decomposition. It would be interesting and challenging to apply the nested decomposition algorithm to other system-defense problems, too.

In **Chapter V** we showed how our approach for solving deterministic system-interdiction problems can be extended to solve a shortest-path network-interdiction problem where interdiction success is uncertain, and the interdictor wishes to maximizes the average length of the post-interdiction shortest path. Even this "simple" stochastic scenario is much more difficult to solve exactly compared to the deterministic analog because evaluating the expected shortest-path length, given an interdiction plan, requires

exponential work.

To deal with the complexity of the stochastic problem we devised several decomposition algorithms that approximate the expected length of the shortest path after interdiction. In these algorithms, the subproblems involve only at a subset of the possible outcomes, or aggregate several outcomes together, so that the number of scenarios considered by the subproblem is manageable. Those approximations yield upper bounds on the optimal objective value. A lower bound can be found, when computationally feasible, by calculating the exact objective value (i.e., by considering all possible scenarios) for one feasible interdiction plan. A good feasible interdiction plan (one that is likely to give a good lower bound) is often suggested by the near-exact solution of the approximation.

Our limited computational experience includes only the basic Benders decomposition for the exact and approximating algorithms. Unfortunately, the subproblems for any of these algorithms generate "flat" cuts, and such cuts cause the master problem to require more iterations to converge compared to analogous deterministic problems. Our computational tests show that all algorithms can have similar running times for the same optimality gap. However, with increased interdiction resources, the approximations are likely to outperform the exact algorithm. Further programming work is needed to check the effectiveness of the integrality cuts and the other basic decomposition algorithms in this stochastic scenario; these techniques might help compensate for the problematic flat cuts.

Our model assumes uncertainty only with respect to the success of each interdiction attempt, but other types of uncertainty might be important as well. For instance, in a shortest-path interdiction problem, the interdictor might not know the exact length of certain arcs, the exact location of the source and target nodes, etc. More work is needed to accommodate these variations of the model.

Among the approximation algorithms, of special interest is an algorithm that finds the shortest paths with respect to the most likely scenarios and uses those to approximate (and bound) the shortest path with respect to all other outcomes. This algorithm seems to give the best results but correctness is only proven for the case in which all arc interdiction-success probabilities are equal.

The approximation algorithms take advantage of the special structure of the stochastic network-interdiction problem, in particular, the fact that first-level variables (interdiction decisions) affect only the cost of the second-level activities. (In a shortest path problem the cost of traversing an arc is the arc's length). However, the same structure can be found in other stochastic programming problems. Consider a stochastic programming problem of the form

$$\max_{\mathbf{x} \in X} \quad E[f(\mathbf{x})]$$

$$\text{where } f(\mathbf{x}) = \max_{\mathbf{y} \in Y} \quad \mathbf{c}^T \mathbf{y}$$

$$s.t. \quad A^T \mathbf{y} \le \mathbf{d} + \widetilde{B} \mathbf{x}$$

where $\widetilde{B} = \text{diag}(\widetilde{\mathbf{b}})$. For instance, this might be a production problem where:

(a)   The variables **x** are binary decisions regarding which markets to expand.

(b)   The demand at facility j is the original demand d_j plus the stochastic result of the market expansion $\tilde{b}_j x_j$. ($x_j = 0$ implies no change in market j.)

(c)   All other parameters (production capabilities and costs, shipping costs, etc.) are deterministic.

If we take the dual of the inner maximization, we obtain a max-min problem with the same structure as the stochastic system-interdiction problems we have solved. Thus, this problem can be solved with the approximation algorithms we have devised.

This thesis contributes mostly to the areas of system interdiction and defense, but our techniques may be helpful in other applications solved by Benders decomposition including certain stochastic-programming problems. Our results should provide ample opportunities for further research in all of these areas.

LIST OF REFERENCES

R. K. Ahuja, T. L. Magnanti and J. B. Orlin, 1993. *Network Flows*, Prentice Hall, Englewood Cliffs, New Jersey.

C. Alexopoulos, 1997. "State space partitioning methods for stochastic shortest path network," *Networks* **30**, 9-21.

G. Anandalingam and V. Apprey, 1991. "Multi-level programming and conflict resolution," *European Journal of Operations Research* **51**, 233-247.

G. Anandalingam and D. White, 1990. "A solution method for the linear static Stackelberg problem using penalty functions," *IEEE Transactions on Automatic Control* **35**, 1170-1173.

M. O. Ball, B. L. Golden and R. V. Vohra, 1989. "Finding the most vital arcs in a network," *Operations Research Letters* **8**, 73-76.

J. Bard, 1984. "Optimality conditions for the bi-level programming problem," *Naval Research Logistics Quarterly* **31**, 13-26.

J. Bard, 1991. "Some properties of the bi-level programming problem," *Journal of Optimization Theory and Applications* **68**, 371-378.

J. Bard and J. Falk, 1982. "An explicit solution to the multi-level programming problem," *Computers and Operations Research* **9**, 77-100.

J. Bard and J. Moore, 1990. "A branch and bound algorithm for the bi-level programming problem," *SIAM Journal on Scientific and Statistical Computing* **11**, 281-292.

J. Bard and J. Moore, 1992. "An algorithm for the discrete bi-level programming problem," *Naval Research Logistics* **39**, 419-435.

E. M. L. Beale, 1959. "On quadratic programming," *Naval Research Logistic Quartely* **6**, 227-243.

J. E. Beasley, 1990. "A lagrangian heuristic for set-covering problems," *Naval Research Logistics* **37**, 151-164.

O. Ben-Ayed, 1993. "Bi-level linear programming," *Computers and Operations Research* **20**, 485-501.

O. Ben-Ayed, D. Boyce and C. Blair, 1988. "A general bi-level linear programming formulation of the network design problem," *Transportation Research* **22 B**, 311-318.

O. Ben-Ayed and C. Blair, 1990. "Computational difficulties of bi-level linear programming," *Operations Research* **38**, 556-560.

J. F. Benders, 1962. "Partioning procedures for solving mixed integer variables programming problems," *Numerische Mathematik* **4**, 238-252.

W. Bialas and M. Karwan, 1984. "Two-level linear programming," *Management Science* **30**, 1004-1020.

J. Bracken and J. McGill, 1973. "Defense applications of mathematical programs with optimization problems in the constraints," *Operations Research* **22**, 1086-1096.

G. G. Brown, G. W. Graves and M. D. Honczarenko, 1987. "Design and operation of a multicommodity production/distribution system using primal goal decomposition," *Management Science* **33**, 1469-1480.

W. Candler and R. Townsley, 1993. "A linear two-level programming problem," *Computers and Operations Research* **9**, 59-76.

A. Caprara, M. Fischetti and P. Toth, 1996. "A heuristic algorithm for the set covering problem," in *Lecture Notes on Computer Science, Vol. 1084, Integer Programming and Combinatorial Optimization*, W. H. Cunningham, S. T. McCormick and M. Queyranne, editors, Springer-Verlag, Berlin, 72-81.

M. S. Chern and K. C. Lin, 1995. "Interdicting the activities of a linear program – A parametric analysis," *European Journal of Operational Research* **86**, 580-591.

H. W. Corely and D. Y. Sha, 1982. "Most vital links and nodes in weighted networks," *Operation Research Letters* **1**, 157-160.

K. J. Cormican, 1995. "Computational methods for deterministic and stochastic network interdiction problems," Masters Thesis, Naval Postgraduate School, Monterey, California.

K. J. Cormican, D. P. Morton and R. K. Wood, 1998. "Stochastic network interdiction," *Operations Research* **46,** 184-197.

J. Falk, 1973. "A linear max-min problem," *Mathematical Programming* **5**, 169-188.

J. Falk and J. Liu, 1995. "On bi-level programming, Part I: General nonlinear case," *Mathematical Programming* **70**, 47-72.

G. S. Fishman, 1985. "Estimating network characteristics in stochastic activity network," *Management Science* **31**, 579-593.

J. Fortuny-Amat and B. McCarl, 1981. "A representation and economic interpretation of a two-level programming problem," *Journal of Operational Research Society* **32**, 783-792.

D. R. Fulkerson and G. C. Harding, 1977. "Maximizing the minimum source-sink path subject to a budget constraint," *Mathematical Programming* **13**, 116-118.

R. S. Garfinkel and G. L. Nemhauser, 1972, *Integer Programming*, John Wiley & Sons, New York.

M. Gendreau, P. Marcotte and G. Savard, 1996. "A hybrid tabu-ascent algorithm for the linear bi-level programming problem," *Journal of Global Optimization* **8**, 217-233.

A.M. Geoffrion and G. W. Graves, 1974. "Multicommodity distribution system design by Benders decomposition," Management Science **20**, 822-844.

P. M. Ghare, D. C. Montgomery and T. M. Turner, 1971. "Optimal interdiction policy for a flow network," Naval Research Logistics Quarterly **18**, 37-45.

B. Golden, 1978. "A problem in network interdiction," *Naval Research Logistics Quarterly* **25**, 711-713.

M. Grötschel, C. Monma and M. Stoer, 1992. "Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints," *Operation Research* **40**, 309-330.

P. Hansen, B. Jaumard and G. Savard, 1992. "New branch-and-bound rules for linear bilevel programming," *SIAM Journal on Scientific and Statistical Computing* **13**, 1194-1217.

ILOG 1997. *Using the CPLEX callable library version 5.0*. ILOG Inc. CPLEX Division, Incline Village, Nevada.

J. J. Judice and A. M. Faustino, 1992. "A sequential LCP method for bilevel linear programming," *Annals of Operation Research* **34**, 89-106.

P. Kall, A. Ruszczyński and K. Frauendorfer, 1988. "Approximation techniques in stochastic programming," in Y. Ermoliev and R. J. Wets (eds.) *Numerical Tecniques for Stochastic Programming*, Springer-Verlag, Berlin, 33-64.

N. Katoh, T. Ibaraki and H. Mine, 1982. "An efficient algorithm for the k shortest simple paths," *Networks* **12**, 411-427.

L. LeBlanc and D. Boyce, 1986. "A bilevel programming algorithm for exact solution of the network design problem with user-optimal flows," *Transportation Research* **20B**, 259-265.

Y. Liu and T. Spencer, 1995. "Solving a bi-level linear program when the inner decision maker controls few variables," *European Journal of Operational Research* **81**, 644-651.

Z.-Q. Luo and J.-S. Pang and D. Ralph, 1996. *Mathematical Programs with Equilibrium Constraints*, Cambridge University Press, Cambridge, U.K.

A. W. McMasters and T. M. Mustin, 1970. "Optimal interdiction of a supply network," *Naval Research Logistic Quarterly* **17**, 261-268.

K. Malik, A. K. Mittal and S. K. Gupta, 1989. "The *k*-most vital arcs in the shortest path problem," *Operation Research Letters* **8**, 223-227.

D. Medhi, 1994. "A unified approach to network survivability for teletraffic networks: models, algorithms and analysis," *IEEE Transactions on Communication* **42**, 534-548.

J. Moore and J. Bard, 1990. "The mixed integer linear bi-level programming problem," *Operations Research* **38**, 911-921.

D. P. Morton and R. K. Wood, 1999. "Restricted-recourse bounds for stochastic linear programming," *Operation Research*, to appear.

G. L. Nemhauser and L. A. Wolsey, 1988. Integer and *Combinatorial Optimization*, Wiley-Interscience, New York.

H. Onal, 1993. "A modified simplex approach for solving bi-level programming problems," *European Journal of Operational Research* **67**, 126-135.

M. Simaan and J.B. Cruz, 1973. "On the Stackelberg strategy in nonzero-sum games," *Journal of Optimization Theory and Applications* **11**, 533-555.

H. A. Taha, 1975. *Integer Programming*, Academic Press, New York.

H. Vaish and C. M. Shetty, 1977. "A cutting plane algorithm for the bilinear programming problem," *Naval Research Logistic Quarterly* **24**, 83-94.

L. Vicente and P. Calamai, 1994. "Bi-level and multilevel programming: A bibliography review," *Journal of Global Optimization* **5**, 291-306.

L. Vicente, G. Savard and J. Judice, 1996. "Discrete linear bi-level programming problem," *Journal of Optimization Theory and Applications* **89**, 597-614.

A. Washburn and K. Wood, 1994. "Two-person zero-sum games for network interdiction," *Operations Research* **43**, 243-251.

U. Wen and S. Hsu, 1991. "Linear bi-level programming problems - a review," *Journal of the Operational Research Society* **42**, 125-133.

U. Wen and Y. Yang, 1990. "Algorithms for solving the mixed integer two-level linear programming problem," *Computers and Operations Research* **17**, 133-142.

The White House, Executive Order 13010, July 15, 1996.

R. K. Wood, 1993. "Deterministic network interdiction," *Mathematical and Computer Modeling* **17**, 1-18.

APPENDIX A. BI-LEVEL LINEAR PROGRAMMING

The system interdiction problem is a min-max, mixed integer, bi-level linear program. In this appendix we introduce the definitions of a general bi-level linear problem and its max-min and mixed-integer variants. The focus of the discussion is on existing algorithms for bi-level problems, and their applicability to the special structure of the system interdiction problem.

A.   BI-LEVEL LINEAR PROGRAMMING

The general *Bi-Level Linear Program* (BLLP) has attracted much attention in the last 30 years (e.g., the literature survey in Vicente and Calamai 1994, the reviews in Ben-Ayed 1993, and Wen and Hsu 1991). Many algorithms have been suggested to solve the BLLP and the model has been applied to a number real-world problems. Nonlinear cases are treated in a few papers (e.g., Falk and Liu 1995) and recently, the more general case, where part of the variables are set as a function of the others through any type of equilibrium constraints, is explored too (Luo, Pang and Ralph 1996).

A bi-level program considers two decision-makers, or *players*, who may be competitive. Each player controls some activities and wishes to optimize his objective function, which is a function of all the activities, including those that the second player controls. The problem can be viewed as a non-zero-sum game where one of the players, the leader, plays first. Due to common constraints, the actions of the leader influence the feasible region of the second player, the follower. In his turn, the follower optimizes his objective function, in view of the decisions of the leader, but independent of the leader's objective function. We assume perfect information, that is, the leader knows the objective function and the constraints of the follower and hence can predict the follower's

reaction to any decision he makes.

The mathematical programming formulation of BLLP is:

[BL]
$$\min_{\mathbf{x} \in X, \mathbf{y} \in Y(\mathbf{x})} \quad \mathbf{c}_1^T \mathbf{x} + \mathbf{c}_2^T \mathbf{y}$$

where $\quad X \quad = \{\mathbf{x} \mid D\mathbf{x} \le \mathbf{d}, \mathbf{x} \ge 0\}$, and

$$Y(\mathbf{x}) = \{\mathbf{y} \mid \mathbf{y} \in \arg\min \quad \mathbf{c}_3^T \mathbf{y}$$

$$\text{s.t.} \quad A\mathbf{y} \le \mathbf{b} - B\mathbf{x}$$

$$\mathbf{y} \ge 0 \quad\quad \}$$

Note that there is no need for a term like $\mathbf{c}_4^T \mathbf{x}$ in the follower's program because $\mathbf{x}$ is a parameter there, not a variable.

$Y(\mathbf{x})$ is called the follower's *rational reaction set*. It is assumed that $Y(\mathbf{x})$ is non-empty for all $\mathbf{x} \in X$. The *inducible region* is defined as $IR = \{ (\mathbf{x},\mathbf{y}) \mid D\mathbf{x} \le \mathbf{d}, \mathbf{x} \ge 0, \mathbf{y} \in Y(\mathbf{x}) \}$. With this notation, we can rewrite [BL] as

[BL1] $\quad\quad \min\{\mathbf{c}_1^T \mathbf{x} + \mathbf{c}_2^T \mathbf{y} \mid (\mathbf{x},\mathbf{y}) \in IR\}$.

The feasible region to [BL1] may not be convex. Therefore, the problem can be difficult to solve and may have local optima. However, let S be the feasible region of [BL], i.e., $S = \{ (\mathbf{x},\mathbf{y}) \mid D\mathbf{x} \le \mathbf{d}, A\mathbf{y} + B\mathbf{x} \le \mathbf{b}, \mathbf{x} \ge 0, \mathbf{y} \ge 0\}$. Then, there is a solution of the problem that is a vertex of S (Bard 1984). As we shall see, many algorithms try to take advantage of that result by performing an implicit search of all possible solutions without enumerating all extreme points of S.

B. THE LINEAR MIN-MAX PROBLEM

[LSIP] without the binary constraints is a special case of [BL] with $c_1 = 0$ and $c_2 = -c_3$. It is also referred to as the *Linear Min-Max Problem* (LMN):

[LMN] $\min\limits_{x \in X, y \in Y(x)} \quad c^T y$

where $X \quad = \{x \mid Dx \le d, x \ge 0\}$, and

$Y(x) \quad = \{y \mid Ay \le b - Bx, y \ge 0\}$.

Remark: In [BL], If $Y(x)$ is not always a singleton, and $c_2^T y$ might not have the same value for all $y \in Y(x)$, then a solution to [BL] may not exist (Bard 1991). However, in [LMN], $c_1 \equiv 0$ and so $c_2^T y$ has the same value for all $y \in Y(x)$. Thus, when $Y(x)$ is bounded for all $x \in X$, an optimal bounded solution for [LMN] must exist.

LMN is equivalent to a structured quadratic problem. Hence, any nonlinear algorithm is a candidate solution method. However, this quadratic objective formulation is non-convex and the problem is, consequently, difficult to solve. To see the equivalence, take the dual of the follower's problem in [LMN] (**w** are the dual variables) to obtain:

[LMN1] $\min\limits_{x,w} \quad (b - Bx)^T w = w^T b - w^T Bx$

s.t. $Dx \quad \le d$

$A^T w \ge c$

$x \ge 0, w \ge 0$.

C. THE BI-LEVEL MIXED-INTEGER PROBLEM

The problem we are interested in, system interdiction, falls into the category of the *Bi-Level Mixed Integer Program* (BLMIP). Let S_X and S_Y represent the binary, and/or integer and/or non-negativity restrictions on the values of the variables $\mathbf{x}$ and $\mathbf{y}$, respectively. Then, the formulation of BLMIP is the same as [BL] except that $\mathbf{x} \in S_X$ replaces $\mathbf{x} \geq \mathbf{0}$ and $\mathbf{y} \in S_Y$ replaces $\mathbf{y} \geq \mathbf{0}$:

$$[\text{BLMIP}] \quad \min_{\mathbf{x} \in X, \mathbf{y} \in Y(\mathbf{x})} \quad \mathbf{c}_1^T \mathbf{x} + \mathbf{c}_2^T \mathbf{y}$$

$$\text{where} \quad X = \{\mathbf{x} \mid D\mathbf{x} \leq \mathbf{d}, \mathbf{x} \in S_X\}, \text{ and}$$

$$Y(\mathbf{x}) = \{\mathbf{y} \mid \mathbf{y} \in \arg\min \quad \mathbf{c}_3^T \mathbf{y}$$

$$\text{s.t.} \quad A\mathbf{y} \leq \mathbf{b} - B\mathbf{x}$$

$$\mathbf{y} \in S_Y \quad \}$$

In the system interdiction scenario, $S_X \subset \{0,1\}^n$ while $S_Y \subset R^m$ and closed (i.e., all the follower's variables are continuous). The general definition of BLMIP is more flexible, however.

D. APPLICATIONS

The ability of the BLLP model to represent decentralized decision processes, where several different objective functions are incorporated simultaneously, has attracted practitioners during the last 30 years. However, the difficulty in solving BLLP constrains the number of actual applications and few models have passed beyond theoretical formulation. Furthermore, in a recent review of the BLLP literature, it is stated that "the overwhelming majority of real-world problems are formulated and solved as single level programs even when they are virtually bi-level (Ben-Ayed 1993)."

Most of the models and work in the area of involve BLLPs with economic interpretations (Fortuny-Amat and McCarl 1981). The upper level (leader) is the government, or the head of an organization, that controls resources, or policy measures such as prices, taxes and subsidies. The lower level (follower) is the private sector, or the lower level of the organization, which optimizes its own objective function, after the higher level sets the rules. Actual work in this area includes agricultural planning, regulation of the oil industry and the imperfect cartel in the international coal market. For a list of references see Vicente and Calamai (1994) and Ben-Ayed (1993).

The BLLP model has also been applied to network-design problems for transportation and communication networks. The two-level formulation takes into account the reaction of users to the improvements made in the system and hence can lead to a better plan for the system manager (e.g., LeBlanc and Boyce 1986, Ben-Ayed, Boyce and Blair 1988).

As one might expect, military models involving worst-case analysis through max-min formulations were popular during the arms race (e.g., Bracken and McGill 1973). However, the models, based on weapons attrition theory, are non-linear, and typically the leader's activities do not change the follower's feasible region. Therefore the solution approach taken there is not applicable to LSIP.

In the LMN framework, we are familiar with two military-oriented applications:

(a)   Max-flow network interdiction (e.g., Wood 1993).

(b)   Shortest-path network interdiction (Golden 1978, Fulkerson and Harding 1977).

In **Chapter II** we discuss in details a version of this problem, in which interdiction decisions are binary, and not continuous, variables.

E.   ALGORITHMS FOR LMN AND BLLP

Currently, few algorithms exist that take advantage of the special structure of LMN. On the other hand, any of the many algorithms that were suggested for BLLP can be used to solve LMN as well (Bard and Falk 1982). These algorithms are of interest because:

(a) We can relax the binary constraints of LSIP if all the extreme points of the leader's feasible region are binary. Thus, in some cases, an algorithm for BLLP (or LMN) solves LSIP directly. Also,

(b) As we shall see later, most of the existing algorithms for mixed-integer problems like LSIP incorporate a continuous-BLLP algorithm to solve sub-problems.

Some of the BLLP algorithms are not well-suited to solving LMNs. As we said earlier, the LMN is a special case of BLLP with $c_2 = -c_3$ and $c_1 = 0$. The correlation (in the usual statistical meaning) between c_2 and c_3 is therefore -1 for the LMN. Significantly, most of the existing algorithms for BLLP are *positive* in the sense that the algorithm works best when there is a strong positive correlation between c_2 and c_3, i.e., when the objective functions of the leader and the follower are similar, with respect to the follower's variables. A positive algorithm is likely to have poor performance when applied to LMN. To solve LMN, we shall use a *non-positive* algorithm, i.e., an algorithm that performs well when there is a strong negative correlation between c_2 and c_3.

While describing the existing algorithms for LMN and BLLP next, we classify each one as positive or non-positive. The classification is based on algorithmic structure and is by no means precise, but, in some of the cases, reported computational experience

- 116 -

also supports our classification. However, unless mentioned specifically, experience with large problems has not been reported.

1. Implicit Enumeration of Possible Bases

Recall that S is the joint feasible region of the two levels in [BL] and that there is an optimal solution at a vertex of S. A few algorithms try to use this result by implicit enumeration of the bases of the polytope that defines S. Usually, the algorithm starts by letting the leader control all the variables in order to find his optimal solution over the vertices of S, regardless of the follower's objective. Fixing the leader's variables, the algorithm solves the follower's problem to determine if the solution is in the inducible region. If it is, we have reached the optimal solution. Otherwise, we can conclude that the basis, or the vertex, that the leader chose is not optimal. Several approaches have been suggested to continue the search over S's vertices:

(a) A branch-and-bound process can be used where each new branch is a sub-problem with one of the basic (and positive) variables from the leader's current basis forced to 0. This algorithm was originally suggested for LMN (Falk 1973).

(b) One variant examines only the follower's bases (Candler and Townsley 1982). Strictly speaking, this algorithm is polynomial when the follower controls a constant number of variables (Liu and Spencer 1993), but it becomes computationally unusable when the number of those variables grows to even modest levels.

(c) The "Kth-best" algorithm (Bialas and Karwan 1982) repeatedly finds solutions for the leader, regardless of the follower's objective, until the Kth best happens to be in the inducible region.

All of the algorithms just mentioned are positive algorithms because the leader's

objective is favored and in each major step the leader controls all the variables. These algorithms cannot be efficient in case of large LMN or LSIP problems.

2. Cutting-Plane Algorithm (Vaish and Shetty 1977)

This algorithm works on the [LMN1] formulation. First it iterates (with appropriate degeneracy-prevention rules) between the $\mathbf{x}$ and $\mathbf{w}$ variables until a stationary point, that may only be only locally optimal, is achieved. Then it computes how far it can move from the extreme point in any feasible direction (with respect to the leader's variables) without improving the objective function, $\mathbf{w}^T\mathbf{b} - \mathbf{w}^T B\mathbf{x}$, cuts this part from X, and starts again. This algorithm works on the LMN problem, but not the BLLP. (BLLP doesn't fit into formulation [LMN1].) Hence, we classify it as a non-positive algorithm.

3. Branch-and-Bound on KKT Complimentary Conditions (Bard and Moore 1990)

The "BB-KKT algorithm" replaces the follower's problem with his KKT optimality conditions. This way, the BLLP is converted to a single-level problem, and a branch-and-bound procedure is used to implicitly examine all combinations of the non-linear complementary slackness conditions. Let $\mathbf{w}$ be the vector of dual variables associated with the constraints $A\mathbf{y} + B\mathbf{x} \leq \mathbf{b}$. Then, the KKT-formulation of BLLP is:

$$[\text{BL2}] \qquad \min_{\mathbf{x,y,w}} \quad \mathbf{c}_1{}^T\mathbf{x} + \mathbf{c}_2{}^T\mathbf{y}$$

$$\text{s.t.} \quad D\mathbf{x} \qquad\qquad \leq \mathbf{d}$$

$$A\mathbf{y} + B\mathbf{x} \qquad \leq \mathbf{b}$$

$$A^T\mathbf{w} \qquad\qquad \geq \mathbf{c}$$

$$(A^T\mathbf{w} - \mathbf{c}_3)^T\mathbf{y} \quad = 0$$

$$(A\mathbf{y} + B\mathbf{x} - \mathbf{b})^T\mathbf{w} = 0$$

$$\mathbf{x} \geq 0, \quad \mathbf{y} \geq 0, \quad \mathbf{w} \geq 0.$$

Random problems with 40 constraints and 100 variables, 40 of them controlled by the follower, were solved by Bard and Moore, in 300 CPU seconds on average, on an IBM 3081-D. The main factor that increases the CPU time is the number of variables controlled by the follower. This result is expected since this is clearly a positive algorithm: In any relaxed iteration of the branch-and-bound process, the "BB-KKT" algorithm controls the leader's and follower's variables together.

4.   Penalty on Duality Gap (Anandalingam and White 1990)

Here, the BLLP is transformed into a single-level program by replacing the follower's problem with a penalty on his duality gap (or equivalently, with a penalty on the complimentary slackness conditions in formulation [BL2]). The reformulation of BLLP is:

$$[BL3] \quad \min_{\mathbf{x}, \mathbf{y}, \mathbf{w}} \quad \mathbf{c}_1^T \mathbf{x} + \mathbf{c}_2^T \mathbf{y} - k(\mathbf{c}_3^T \mathbf{y} - \mathbf{w}^T(\mathbf{b} - B\mathbf{x}))$$

$$\text{s.t.} \quad D\mathbf{x} \quad \leq \mathbf{d}$$

$$A\mathbf{y} + B\mathbf{x} \quad \leq \mathbf{b}$$

$$\mathbf{w}^T A \quad \geq \mathbf{c}_3$$

$$\mathbf{x} \geq \mathbf{0}, \mathbf{y} \geq \mathbf{0}, \mathbf{w} \geq \mathbf{0}.$$

Let the optimal solution of [BL3] be $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{w}^*)$. $\mathbf{x}^*$ is feasible in the leader's problem. Moreover, for k sufficiently large, the duality gap must be zero, so we must have $\mathbf{y}^* \in Y(\mathbf{x}^*)$. Two different algorithms have been suggested to solve [BL3];

Let $\theta(\mathbf{w}) = \min_{\mathbf{x}, \mathbf{y}} \{ \mathbf{c}_1^T \mathbf{x} + \mathbf{c}_2^T \mathbf{y} - k(\mathbf{c}_3^T \mathbf{y} - \mathbf{w}^T(\mathbf{b} - B\mathbf{x})) \mid D\mathbf{x} \leq \mathbf{d}, A\mathbf{y} + B\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq$

$\mathbf{0}, \mathbf{y} \geq \mathbf{0}\}$ then, [BL3] is equivalent to $\min_{\mathbf{w}} \{ \theta(\mathbf{w}) \mid \mathbf{w}^T A \geq \mathbf{c}_3, \mathbf{w} \geq \mathbf{0} \}$, which is a difficult

concave minimization problem. Anandalingam and Apprey (1991) solves this problem with a successive underestimation method proposed by Falk and Hoffman (1976).

A second way to solve [BL3] is through a "modified simplex algorithm" (Onal 1993), which is a quadratic programming algorithm (Beale 1959) that seems to become simpler in the special setting of [BL3]. The algorithm maintains a basic feasible solution and, in each "simplex" iteration, it evaluates the partial derivatives of the objective function with respect to the non-basic variables (through a relatively complex matrix calculation). Then, it pivots and chooses the entering variable in the usual way. In each major iteration, a local optimum might be found, and upper bound (UB) is updated and a cut, $\mathbf{c}_1^T \mathbf{x} + \mathbf{c}_2^T \mathbf{y} \leq UB - \varepsilon$, is added to the problem. The problem is re-solved until no feasible solution is found; then we have an ε-optimal global, solution.

It is not clear whether these two algorithms should be classified as positive or non-positive. In both algorithms, the primal variables of the leader and the follower are controlled simultaneously, and this is a positive approach. However, both algorithms have a "correction" step where only the follower's dual variables are considered. That might be considered as a follower's step.

5.   Complementarity Approach (Judice and Faustino 1992)

The idea of transforming formulation [BL2] into a parametric linear complementarity problem was suggested by Bialas and Karwan (1984) but their algorithm isn't guaranteed to converge to an optimal solution (Ben-ayed and Blair 1990). In 1988, Judice and Faustino suggested a modification that is guaranteed to converge to an ϵ-optimal solution. The algorithm was tested on random problems with up to 150 constraints and 250 variables, and was shown to have relatively good performance (Hansen, Jaumard and Savard 1992). However, the performance of the algorithm degrades significantly when tested on problems with "conflicting" objectives. This is a clear evidence that the algorithm is positive.

6.   Variable Elimination (Hansen, Jaumard and Savard 1992)

This algorithm performs branch-and-bound on the constraints of the follower's problem. At each node on the enumeration tree, one more of these constraints is forced to bind and one of the follower's variables is eliminated. Then, the algorithm lets the leader control the follower's remaining unfixed variables and the leader's problem is solved to obtain a local lower bound on the leader's optimal solution. This local bound may not be global because some of the follower's variables have already been set. If this local lower bound is worse than the bound from a feasible solution, this node can be

eliminated. The algorithm is implemented using a depth-first search of the enumeration tree and each time the search reaches a leaf (all follower's variables are set), the leader's problem is solved, a feasible solution is exposed and the global upper bound on the follower's optimal solution is updated, if appropriate.

The algorithm was extensively tested on problems with up to 150 constraints and 250 variables. It was shown to be much better than any other tested algorithm, including branch-and-bound on the KKT complimentary conditions (Bard and Moore 1990) and the Judice-Faustino algorithm. However, like those two algorithms, certain random problems require ten times more computational effort than other problems, and sensitivity to the number of follower variables is observed. Furthermore, a special set of tests indicates that the running time decreases when the leader's and follower's objective functions become more similar. Those results indicate that this is a positive algorithm. Indeed, the lower bound on the follower's objective used for fathoming will rarely be effective in the min-max case.

7. **A Hybrid Tabu-Ascent Heuristic** (Gendreau, Marcotte and Savard 1996).

This algorithm attempts to solve BLLP to near-optimality by a combination of a few heuristic methods. After a feasible solution is found, the algorithm iterates between a local-ascent search and a tabu search. The first technique is used to find an optimum that might be local, and the second is used to move away from a local optimum, to an area where a solution better than the last local optimum may exist. When the tabu search fails to find such an area, the algorithm stops.

The algorithm was tested and compared to the variable elimination algorithm on a set of random problems, on an HP730 workstation. The largest problems had 200 variables, 75 constraints and a relatively dense constraint matrix (25% non-zero). On those problems, the heuristic algorithm used 3-13 minutes of CPU time and found better solutions than the exact algorithm found after 60 minutes of CPU time.

Recall that the variable elimination algorithm has significant variation in running time between problems of the same size, probably because it is a positive algorithm. The heuristic algorithm has a much more stable running time. Hence, we can guess that the heuristic is using a non-positive approach.

F. ALGORITHMS FOR THE BLMIP

Only a few papers have been written concerning solution procedures for the bi-level mixed integer programming (BLMIP).

1. Parameterized integer program (Bard and Moore 1992)

The algorithm works only when both $\mathbf{x}$ and $\mathbf{y}$ are binary. It works by solving instances of the following formulation

$$[\text{BLMIP1}] \quad \max_{\mathbf{x},\mathbf{y}} \quad \mathbf{c}_3^T \mathbf{y}$$

$$\text{s.t.} \quad D\mathbf{x} \quad \leq \mathbf{d}$$

$$A\mathbf{y} + B\mathbf{x} \quad \leq \mathbf{b}$$

$$\mathbf{c}_1^T \mathbf{x} + \mathbf{c}_2^T \mathbf{y} \leq \alpha$$

$$\mathbf{x} \in X, \mathbf{y} \in Y$$

where α is a parameter, equal to the best objective value found so far less 1. (It is assumed that all objective coefficients are integer.)

The algorithm implicitly enumerates the enumeration tree associated with the variables **x**. In each iteration, some of those variables are set to 0 or 1 and the remaining problem is solved as a standard MIP. When the MIP is infeasible, the branch is deleted. Otherwise, **x** is fixed and the follower's problem is solved to obtain a feasible solution to the original problem, [BLMIP]. If the two solutions are the same, there is no need to develop this branch anymore. Otherwise the enumeration process continues by restricting **x** further. The algorithm uses specific branching and backtracking rules that were optimized with respect to a set of random problems.

Computational results with random problems with up to 45 binary variables were reported. On average it took 100-150 CPU seconds on an IBM 3081-D to solve problems with 30 **x** variables and 15 **y** variables. Wide variations in the algorithm's running time and sensitivity to the number of variables under the follower's control were observed. Both these results are probably because this is a positive algorithm. If we substitute the LSIP conditions, $\mathbf{c}^1 = \mathbf{0}$ and $\mathbf{c}^3 = -\mathbf{c}^2$, then [BLMI1] would become almost useless and the algorithm might enumerate all possible values for **x**.

2. "Classical" branch-and-bound (Moore and Bard 1990)

The algorithm solves the mixed integer bi-level program similar to the way branch-and-bound is used to solve MIPs. At every node, a relaxed BLLP is solved and the enumeration tree is developed and fathomed with appropriate fathoming rules. When all the leader's variables are continuous, as in LSIP, the three "regular" fathoming rules (infeasiblity, new integer solution and objective value worse then a known integer solution) are valid and used in the usual way. When all or part of the follower's variables are restricted to integer values, the fathoming rules must be modified, however.

This algorithm's performance hinges on the sub-algorithm that is used to solve the relaxed BLLP problems (which are NP-hard.) Originally the algorithm was implemented with the "BB-KKT algorithm", which is positive, and not appropriate for solving LSIP. The algorithm could work with other sub-algorithms as well, but, recall that we have found no non-positive algorithm for BLLP that has been tested on large problems.

3.   Binary search algorithm (Wen and Yang 1990)

As in LSIP, this algorithm assumes binary leader variables and continuous follower variables. However, the algorithm doesn't assume the min-max case, and allows independent objective functions for the leader and the follower.

The algorithm implicitly examines all possible values of the leader's variables through a branch-and-bound process. At each node in the enumeration tree, a few of the leader's variables are set to 0 or 1 and the algorithm uses an LP (which we won't describe here) to calculate a local lower bound on the leader's objective. This branch is fathomed if the bound is worse than the best solution value found so far. When all the variables x have been set to 0 or 1, the follower's problem is solved to obtain a new feasible solution.

The algorithm is positive, and when it's applied to a problem such as system interdiction, the lower bound from the LP will not be useful at all. Thus, the algorithm might enumerate all possible values for x. Moreover, even on random problems the algorithm's performance is poor and the authors suggest using a heuristic, which we describe next.

4.   Greedy Heuristic (Wen and Yang, 1990)

This heuristic uses a greedy approach. At each iteration, a few of the leader's variables are set to 1 and the follower's problem is solved, as well as the LP mentioned above. Based on the dual variables of these solutions, a "judgment index" for the profit of the leader from setting each one of his variables to 1 is calculated. The variable with the maximum judgment index is set to 1. The algorithm starts with all the variables equal to 0. In each iteration, it sets one more variable to 1, and continues until no more variables can be set without violating feasibility. The heuristic is extremely fast and, on average, achieves better than a 3% optimality gap when tested on random problems. However, it is easy to build an example where this simple greedy heuristic will give arbitrarily bad results.

The heuristic becomes simpler when it applied to LSIP. In every iteration, the follower's problem is solved with the current set of interdicted activities, the "judgment index" of interdicting activity j is u_j times the dual variable of the constraint $y_j \leq u_j$. The activity with the largest judgement index is interdicted, and the process continuous recursively.

5.   Reducing into BLLP (Vicente, Savard and Judice 1996)

The last algorithm suggested so far for solving BLMIP transforms the problem into a regular BLLP. It is shown that if (a) **y** is continuous, (b) **x** is binary, (c) M is sufficiently large, and, (d) $\mathbf{c}_1 \leq \mathbf{0}$ and $\mathbf{c}_2 \leq \mathbf{0}$ (without loss of generality), then [BLMIP] is equivalent to:

[BLMIP2] $\displaystyle\min_{\mathbf{x}\in X,(\mathbf{y},\mathbf{w})\in YW(\mathbf{x})}$ $c_1{}^T\mathbf{x}+c_2{}^T\mathbf{y}+M\mathbf{e}^T\mathbf{w}$

where $X=\{\mathbf{x}\mid D\mathbf{x}\le\mathbf{d},0\le\mathbf{x}\le\mathbf{e}\}$

$YW(\mathbf{x})=\{(\mathbf{y},\mathbf{w})\mid(\mathbf{y},\mathbf{w})\in\displaystyle\arg\max_{\mathbf{y},\mathbf{w}}\quad c_3{}^T\mathbf{y}+\mathbf{e}^T\mathbf{w}$

s.t. $A\mathbf{y}\le\mathbf{b}-B\mathbf{x}$

$\mathbf{w}\ \le\mathbf{x}$

$\mathbf{w}\ \le\mathbf{e}-\mathbf{x}$

$\mathbf{y}\ \in Y\qquad\}$

where $\mathbf{e}$ is a vector of ones with appropriate dimension.

As with other models using penalty terms, a key problem is to determine how large M should be. Vicente *et al.* suggest solving a sequence of BLLPs for increasing values of the penalty term. When the optimal solution in the kth iteration has all leader variables at 0 or 1 we can stop and declare optimality. Otherwise $M=M_k$ is increased to $M_{k+1}=(-c_1{}^T\mathbf{x}_k-c_2{}^T\mathbf{y}_k)/\mathbf{e}^T\mathbf{w}_k$, and the non-integer optimal solution of the kth iteration will give a positive and unattractive objective in the following iteration. Computational experience has not been reported and it is not clear whether computational difficulties are expected (due to the large penalty multiplier).

G.   CONCLUSIONS

The main conclusion of this appendix is that the existing literature on general BLLPs does not suggest suitable algorithms for the special structure of LSIP. We support this statement with the following observations:

(a)   Only three exact algorithms (Bard and Moore 1990, Hansen, Jaumard and Savard 1992, Section 16.3.3 in Shimizu, Ishizuka and Bard 1997) have been tested on

relatively large BLLPs. These algorithms all use a strongly positive approach, which means that they work better when there is strong correlation between the leader's and follower's objective functions. Hence, they would probably not be efficient when applied to min-max problems, like LSIP. Only a few exact algorithms (Vaish and Shetty 1977, Anandalingam and Apprey 1991, Onal, 1993) that may use a non-positive approach exist, and none of those has been tested on large problems.

(b)   Existing algorithms for the mixed integer case, BLMIP, are either positive by themselves (Bard and Moore 1992, Wen and Yang 1990) or are based on a BLLP algorithm as a subroutine (Moore and Bard 1989, Vicente, Savard and Judice 1996). And, as mentioned above, there are no BLLP algorithms that seem to be attractive for solving LSIP.

(c)   None of the existing algorithms, exact or heuristic, is designed to take advantage of the special min-max structure of LSIP. Moreover, the algorithms have typically been tested on "random problems."

We have established that none of the existing algorithms for BLLPs or BLMIPs is really appropriate for LSIP. Therefore, in **Chapters II** and **Chapter III**, we develop three new algorithms specially tailored to LSIP. The algorithms definitely exploit the special structure of LSIP and cannot be applied to a general BLMIP. In **Chapter II**, we demonstrate the effectiveness of the algorithms on a special-case problem, shortest-path network interdiction.

APPENDIX B: THE MORE GENERAL SYSTEM INTERDICTION PROBLEM

Proposition 3.1: *When* **Assumption 3.1** *holds, the basic linear system interdiction*

[LSIP] $\min\limits_{\mathbf{x} \in X} \quad \max\limits_{\mathbf{y} \in Y(\mathbf{x})} \quad \mathbf{c}^T \mathbf{y}$

where $X \quad = \left\{\mathbf{x} \in \{0,1\}^n \,\middle|\, R\mathbf{x} \le \mathbf{r}\right\}$, and

$Y(\mathbf{x}) \ = \left\{\mathbf{y} \,\middle|\, A\mathbf{y} \le \mathbf{b}, 0 \le \mathbf{y} \le U(1-\mathbf{x}), \mathbf{y} \in S\right\}$

is equivalent to the more general linear system-interdiction problem defined as

[LSIP*] $\min\limits_{\hat{\mathbf{x}} \in \hat{X}} \quad \max\limits_{\hat{\mathbf{y}} \in Y(\hat{\mathbf{x}})} \quad \hat{\mathbf{c}}^T \hat{\mathbf{y}} - \hat{\mathbf{x}}^T \hat{V} \hat{\mathbf{y}}$

where $\hat{X} \quad = \left\{\hat{\mathbf{x}} \in \{0,1\}^{\hat{n}} \,\middle|\, \hat{R}\hat{\mathbf{x}} \le \hat{\mathbf{r}}\right\}$, and

$Y(\hat{\mathbf{x}}) = \left\{\hat{\mathbf{y}} \in R^{\hat{m}} \,\middle|\, \hat{A}\hat{\mathbf{y}} \le \hat{\mathbf{b}} - \hat{B}\hat{\mathbf{x}}, 0 \le \hat{\mathbf{y}}, \hat{\mathbf{y}} \in \hat{S}\right\}$

where U=diag($\mathbf{u}$), S and $\hat{S}$ include integer or binary restrictions on elements of $\mathbf{y}$ and $\hat{\mathbf{y}}$ respectively, $A \in R^{k \times m}, V \in R_+^{\hat{n} \times \hat{m}}, \hat{A} \in R^{\hat{k} \times \hat{m}}$ and $B \in R_+^{\hat{k} \times \hat{n}}$.

Remarks:

(a)   The general case, [LSIP*], allows an interdiction by the leader to affect one or more of the follower's available resources and/or the availability and cost of the follower's possible activities.

(b)   The restriction that $B_{ij} \ge 0$ is acceptable, because it is unlikely that an interdiction will relax any of the system's constraints.

Proof:

Set $\hat{A} = \begin{bmatrix} A \\ I \end{bmatrix}$, $\hat{\mathbf{b}} = \begin{bmatrix} \mathbf{b} \\ \mathbf{u} \end{bmatrix}$, $\hat{B} = \begin{bmatrix} \mathbf{0} \\ B \end{bmatrix}$, and $V = [\mathbf{0}]$. Then, [LSIP] takes the form of

[LSIP*].

For the second direction, we first show how to drop out the non-linear term in the objective function of [LSIP*]. Let us introduce n new variables, $\overline{\mathbf{y}} \in R_+^n$, and n new

constraints, $\overline{\mathbf{y}} \geq V\hat{\mathbf{y}} - V(\mathbf{1} - \mathbf{x})$. Set $\hat{\hat{\mathbf{y}}} = \begin{bmatrix} \hat{\mathbf{y}} \\ \overline{\mathbf{y}} \end{bmatrix}$, $\hat{\hat{\mathbf{c}}} = \begin{bmatrix} \hat{\mathbf{c}} \\ -\mathbf{1} \end{bmatrix}$, $\hat{\hat{\mathbf{b}}} = \begin{bmatrix} \hat{\mathbf{b}} \\ V\mathbf{1} \end{bmatrix}$, $\hat{\hat{A}} = \begin{bmatrix} \hat{A} & \mathbf{0} \\ V & -I \end{bmatrix}$,

$\hat{\hat{B}} = \begin{bmatrix} B \\ V \end{bmatrix}$, and $V \equiv [\mathbf{0}]$. These new constraints and the unattractive coefficients of $\overline{\mathbf{y}}$ in

the objective function, guarantee that $\overline{y}_j = \sum_{x_j=1} (V\mathbf{y})_j$ if $x_j = 1$ (recall that $V \in R_+^{\hat{n} \times \hat{m}}$).

Therefore, for every $\mathbf{x} \in X$ the solution of the inner maximization problem stays the same, and so this transformation doesn't change the optimal solution. We conclude that we can assume that $V = [\mathbf{0}]$.

We now show how to rearrange the structure of the constraints in $Y(\hat{\mathbf{x}})$. Let us

introduce $(\hat{k} \times \hat{n} + \hat{n})$ new variables, $\mathbf{z} \in R_+^{(\hat{k} \times \hat{n})}$ and $\overline{\overline{\mathbf{y}}} \in R^{\hat{n}}$. (Note that $\mathbf{z}$ is a vector, not

a matrix.) We add the following constraints: (a) $\overline{\overline{\mathbf{y}}} \leq \mathbf{1} - \hat{\mathbf{x}}$, and (b) $z_{(i-1)\hat{n}+j} \geq B_{ij}(1 - \overline{\overline{y}}_j)$

for $i=1,..,\hat{k}$ and $j=1,..,\hat{n}$. (For simplicity we denote $z_{(i-1)\hat{n}+j}$ by z_{ij}.)

To finish the construction, the original jth constraint, $\sum_{j=1}^{\hat{m}} \hat{A}_{ij}\hat{y}_j \leq \hat{b}_j - \sum_{j=1}^{\hat{n}} \hat{B}_{ij}\hat{x}_j$, is

replaced by $\sum_{j=1}^{\hat{m}} \hat{A}_{ij}\hat{y}_j \leq \hat{b}_j - \sum_{j=1}^{\hat{n}} z_{ij}$. This replacement changes nothing, because:

(a)   If activity j is interdicted ($x_j = 1$), then these constraints and construction set

$\bar{y}_j = 0$ and therefore $z_{ij} \geq B_{ij}$ for i. However, for every interdiction plan $\mathbf{x}$, there

is an optimal solution with $z_{ij} = B_{ij}$ for all i, because $\mathbf{z}$ only tightens constraints.

(b)   In the same way, when activity j is not interdicted ($x_j = 0$), for every interdiction

plan $\mathbf{x}$, there is an optimal solution with $z_{ij} = 0$ for all i.

It is easy to verify that the new construction fits into formulation [LSIP], where

$\bar{y}, \bar{\bar{y}}$, and $\mathbf{z}$ are aggregated into $\mathbf{y}$, and A, U, and $\mathbf{b}$, are defined appropriately. ∎

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center   2
   8725 John J. Klingman Rd., Ste 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library (411 Dyer Rd.)   2
   Naval Postgraduate School
   Monterey, CA 93943-5101

3. Professor R. Kevin Wood Code OR/Wd   10
   Naval Postgraduate School
   Monterey, CA 93943-5002

4. Professor Gerald G. Brown Code OR/Bw   1
   Naval Postgraduate School
   Monterey, CA 93943-5002

5. Professor Alan R. Washburn Code OR/Ws   1
   Naval Postgraduate School
   Monterey, CA 93943-5002

6. Adjunct Professor Wayne P. Hughes Code OR/Hl   1
   Naval Postgraduate School
   Monterey, CA 93943-5002

7. Professor Harold Fredricksen Code MA/Fr   1
   Naval Postgraduate School
   Monterey, CA 93943-5002

8. Major Eitan Israeli   5
   2 Arbel St.
   Reu't, Israel, 71908